University of California
Santa Barbara

Combating Attacks and Abuse in Large Online Communities

A dissertation submitted in partial satisfaction
of the requirements for the degree

Doctor of Philosophy
in
Computer Science

by

Gang Wang

Committee in charge:

Professor Ben Y. Zhao, Chair
Professor Haitao Zheng
Professor Christopher Kruegel

September 2016

The Dissertation of Gang Wang is approved.

Professor Christopher Kruegel

Professor Haitao Zheng

Professor Ben Y. Zhao, Committee Chair

July 2016

Combating Attacks and Abuse in Large Online Communities

Acknowledgements

I would like to thank my advisors Ben Y. Zhao and Haitao Zheng for mentoring me throughout the PhD program. They were always there for me, giving me timely and helpful advice in almost all aspects of both research and life. I also want to thank my PhD committee member Christopher Kruegel for his guidance in my research projects and job hunting. Finally, I want to thank my mentors in previous internships: Jay Stokes, Cormac Herley, Weidong Cui and Helen Wang from Microsoft Research, and Vicente Silveira from LinkedIn. Special thanks to Janet Kayfetz, who has helped me greatly with my writing and presentation skills.

I am very much thankful to my collaborators for their hard work, without which none of this research would have been possible. First and foremost, to the members of SAND Lab at UC Santa Barbara: Christo Wilson, Bolun Wang, Tianyi Wang, Manish Mohanlal, Xiaohan Zhao, Zengbin Zhang, Xia Zhou, Ana Nika, Xinyi Zhang, Shiliang Tang, Alessandra Sala, Yibo Zhu, Lin Zhou, Weile Zhang, Konark Gill, Divya Sambasivan, Xiaoxiao Yu, Troy Steinbauer, Tristan Konolige, Yu Su and Yuanyang Zhang. Second, to the employees at Microsoft: Jack Stokes, Cormac Herley and David Felstead. Third, to Miriam Metzger from the Department of Communications at UC Santa Barbara, and Sarita Y. Schoenebeck from the School of Information at University of Michigan. Fourth, to our collaborators from SecLab at UC Santa Barbara: Gianluca Stringhini, Manuel Egele, Christopher Kruegel and Giovanni Vigna. Finally, to Xiao Wang at Renren Inc., David Freeman at LinkedIn Inc., and Ulas Bardak at Whisper Inc. for sharing data for my research.

Curriculum Vitæ

Gang Wang

Education

2010–2016	Ph.D. in Computer Science, University of California, Santa Barbara.
2006–2010	B.E. in Electronic Engineering, Tsinghua University

Field of Study

Major Field	Computer Science with Prof. Ben Y. Zhao and Prof. Haitao Zheng.

Employment

2010/9–2016/7	Research Assistant, UC Santa Barbara, Santa Barbara, CA.
2014/6–2014/9	Research Internship, Microsoft Research, Redmond, WA.
2012/6–2012/9	Data Scientist Internship, LinkedIn Inc., Mountain View, CA.
2011/6–2011/9	Research Internship, Microsoft Research, Redmond, WA.
2010/3–2010/6	Research Internship, Technicolor Research, Beijing, China.

Publications

MobiSys'16	Defending Against Sybil Devices in Crowdsourced Mapping Services. Gang Wang, Bolun Wang, Tianyi Wang, Ana Nika, Haitao Zheng, Ben Y. Zhao. In Proc. of International Conference on Mobile Systems, Applications, and Services, June 2016.
ICWSM'16	"Will Check-in for Badges": Understanding Bias and Misbehavior on Location-based Social Networks. Gang Wang, Sarita Schoenebeck, Haitao Zheng, Ben Y. Zhao. In Proc. of International AAAI Conference on Web and Social Media, May 2016.
CHI'16	Unsupervised Clickstream Clustering For User Behavior Analysis. Gang Wang, Xinyi Zhang, Shiliang Tang, Haitao Zheng, Ben Y. Zhao. In Proc. of SIGCHI Conference on Human Factors in Computing Systems, May 2016.
CSCW'15	Crowds on Wall Street: Extracting Value from Collaborative Investing Platforms. Gang Wang, Tianyi Wang, Bolun Wang, Divya Sambasivan, Zengbin Zhang, Haitao Zheng, Ben Y. Zhao. In Proc. of ACM conference on Computer-Supported Cooperative Work and Social Computing, March 2015.

FCS'15	The Power of Comments: Fostering Social Interactions in Microblog Networks. Tianyi Wang, Yang Chen, Yi Wang, Bolun Wang, Gang Wang, Xing Li, Haitao Zheng, Ben Y. Zhao. Springer Frontiers of Computer Science, 2015.
IMC'14	Whispers in the Dark: Analysis of an Anonymous Social Network. Gang Wang, Bolun Wang, Tianyi Wang, Ana Nika, Haitao Zheng, Ben Y. Zhao. In Proc. of Internet Measurement Conference, November 2014.
USENIX SEC'14	Man vs. Machine: Practical Adversarial Detection of Malicious Crowdsourcing Workers. Gang Wang, Tianyi Wang, Haitao Zheng, Ben Y. Zhao. In Proc. of USENIX Security Symposium, August 2014.
TON'14	Practical Conflict Graphs in the Wild. Xia Zhou, Zengbin Zhang, Gang Wang, Xiaoxiao Yu, Ben Y. Zhao, Haitao Zheng. ACM Transactions on Networking, 2014.
HotNets'13	On the Validity of Geosocial Mobility Traces. Zengbin Zhang, Lin Zhou, Xiaohan Zhao, Gang Wang, Yu Su, Miriam Metzger, Haitao Zheng, Ben Y. Zhao. In Proc. of Workshop on Hot Topics in Networks, November 2013.
IMC'13	Follow the Green: Growth and Dynamics in Twitter Follower Markets. Gianluca Stringhini, Gang Wang, Manuel Egele, Christopher Kruegel, Giovanni Vigna, Haitao Zheng, Ben Y. Zhao. In Proc. of Internet Measurement Conference, October 2013.
USENIX SEC'13	You are How You Click: Clickstream Analysis for Sybil Detection. Gang Wang, Tristan Konolige, Christo Wilson, Xiao Wang, Haitao Zheng, Ben Y. Zhao. In Proc. of USENIX Security Symposium, August 2013.
DSN'13	Detecting Malicious Landing Pages in Malware Distribution Networks. Gang Wang, Jack Stokes, Cormac Herley, David Felstead. In Proc. of IEEE/IFIP International Conference on Dependable Systems and Networks, June 2013.
SIGMETRICS'13	Practical Conflict Graphs for Dynamic Spectrum Distribution. Xia Zhou, Zengbin Zhang, Gang Wang, Xiaoxiao Yu, Ben Y. Zhao, Haitao Zheng. In Proc. of International Conference on Measurement and Modeling of Computer Systems, June 2013.
WWW'13	Wisdom in the Social Crowd: an Analysis of Quora. Gang Wang, Konark Gill, Manish Mohanlal, Haitao Zheng, Ben Y. Zhao. In Proc. of International World Wide Web Conference, May 2013.
NDSS'13	Social Turing Tests: Crowdsourcing Sybil Detection. Gang Wang, Manish Mohanlal, Christo Wilson, Xiao Wang, Miriam Metzger, Haitao Zheng, Ben Y. Zhao. In Proc. of Network & Distributed System Security Symposium, February 2013.

WWW’12	Serf and Turf: Crowdturfing for Fun and Profit. Gang Wang, Christo Wilson, Xiaohan Zhao, Yibo Zhu, Manish Mohanlal, Haitao Zheng, Ben Y. Zhao. In Proc. of International World Wide Web Conference, April 2012.
MobiCom’11	I am the Antenna: Accurate Outdoor AP Location using Smartphones. Zengbin Zhang, Xia Zhou, Weile Zhang, Yuanyang Zhang, Gang Wang, Ben Y. Zhao, Haitao Zheng. In Proc. of International Conference on Mobile Computing and Networking, September 2011.
HotMobile’11	Privacy, Availability and Economics in the Polaris Mobile Social Network. Christo Wilson, Troy Steinbauer, Gang Wang, Alessandra Sala, Haitao Zheng, Ben Y. Zhao. In Proc. of Workshop on Mobile Computing Systems and Applications, March 2011.
P2PNet’09	Experimental Study on Neighbor Selection Policy for Phoenix Network Coordinate System. Gang Wang, Shining Wu, Guodong Wang, Beixing Deng, Xing Li. In Proc. of Workshop on Peer-To-Peer Networking, October, 2009.

Abstract

Combating Attacks and Abuse in Large Online Communities

by

Gang Wang

Internet users today are connected more widely and ubiquitously than ever before. As a result, various online communities are formed, ranging from online social networks (Facebook, Twitter), to mobile communities (Foursquare, Waze), to content/interests based networks (Wikipedia, Yelp, Quora). While users are benefiting from the ease of access to information and social interactions, there is a growing concern for users' security and privacy against various attacks such as spam, phishing, malware infection and identity theft.

Combating attacks and abuse in online communities is challenging. First, todays online communities are increasingly dependent on users and user-generated content. Securing online systems demands a deep understanding of the complex and often unpredictable human behaviors. Second, online communities can easily have millions or even billions of users, which requires the corresponding security mechanisms to be highly scalable. Finally, cybercriminals are constantly evolving to launch new types of attacks. This further demands high robustness of security defenses.

In this thesis, we take concrete steps towards measuring, understanding, and defending against attacks and abuse in online communities. We begin with a series of empirical measurements to understand user behaviors in different online services and the unique security and privacy challenges that users are facing with. This effort covers a broad set of popular online services including social networks for question and answering (Quora), anonymous social networks (Whisper), and crowdsourced mobile communities (Waze). Despite the differences of specific online communities, our study provides a first look at their user activity patterns based on empirical data, and reveals the need for reliable mechanisms to curate user content, protect privacy, and defend against emerging attacks.

Next, we turn our attention to attacks targeting online communities, with focus on spam campaigns. While traditional spam is mostly generated by automated software, attackers today start to introduce "human intelligence" to implement attacks. This is malicious crowdsourcing (or crowdturfing) where a large group of real-users are organized to carry out malicious campaigns, such as writing fake reviews or spreading rumors on social media. Using collective human efforts, attackers can easily bypass many existing defenses (*e.g.*, CAPTCHA). To understand the ecosystem of crowdturfing, we first use measurements to examine their detailed campaign organization, workers and revenue. Based on insights from empirical data, we develop effective machine learning classifiers to detect crowdturfing activities. In the meantime, considering the adversarial nature of crowdturfing, we also build practical adversarial models to simulate how attackers can evade or disrupt machine learning based defenses.

To aid in this effort, we next explore using user behavior models to detect a wider range of attacks. Instead of making assumptions about attacker behavior, our idea is to model normal user behaviors and capture (malicious) behaviors that are deviated from norm. In this way, we can detect previously unknown attacks. Our behavior model is based on detailed clickstream data, which are sequences of click events generated by users when using the service. We build a similarity graph where each user is a node and the edges are weighted by clickstream similarity. By partitioning this graph, we obtain "clusters" of users with similar behaviors. We then use a small set of known good users to "color" these clusters to differentiate the malicious ones. This technique has been adopted by real-world social networks (Renren and LinkedIn), and already detected unexpected attacks. Finally, we extend clickstream model to understanding more-grained behaviors of attackers (and real users), and tracking how user behavior changes over time.

In summary, this thesis illustrates a data-driven approach to understanding and defending against attacks and abuse in online communities. Our measurements have revealed new insights about how attackers are evolving to bypass existing security defenses today. In addition, our data-driven systems provide new solutions for online services to gain a deep understanding of their users, and defend them from emerging attacks and abuse.

Contents

Chapter 1

Introduction

Internet users today are connected more widely and ubiquitously than ever before. As a result, various online communities are formed ranging from online social networks (Facebook, Twitter, LinkedIn, Snapchat), to content/interests based communities (Wikipedia, Yelp, Quora), to mobile and location based networks (Foursquare, Waze). Online communities are gaining an increasing popularity. As of the first quarter of 2016, the largest online social network Facebook has reached 1.65 billion monthly active users [30] and Twitter's population has reached 310 million [34]. Even the much younger Snapchat is catching up with 100 million daily users around the same time [33]. Online communities not only ease the process of information seeking for users, but also enable more timely and intimate communications among friends, families and even strangers. Meanwhile, the massive content in these online communities are playing a significant role in users' daily life and decision making, ranging from online restaurant reviews in Yelp, to recommendations of cruise lines on TripAdvisor, to financial advice in SeekingAlpha and StockTwits.

Due to their significant influence, online communities have become attractive targets for malicious *attacks*, including opinion manipulation on product reviews [113, 138, 114], political lobbying campaigns [180], and massive distribution of malicious content (social spam, scam and malwares) [77, 88, 206]. Meanwhile, *privacy* is also a growing concern in online communities, as online services are aggressively aggregating user personal information such as real names, email addresses, phone numbers, and social connections. Evidence has shown that dedicated attackers can successfully de-anonymize users based on their online footprints [158, 152]. In addition, with the rise of mobile communities (Foursquare, Uber, Yik Yak), online data is much more closely linked to users' physical activities. This renders even stronger privacy implications since attackers may infer who you are based on where you live or work.

To put these security and privacy issues into context, we now describe the prevalent attacks and abusive behaviors in online communities and discuss the challenges to address them.

Social Spam. A key advantage of spam within online communities is the higher-level of "trust" among social friends. Such trust often gives users the false sense of security making them more likely to engage with the spam content passed from friends. Several studies have shown that links embedded in social spam have a much higher click-through rates than those in email spam [88]. In addition, social spam is usually the first step of more severe attacks such as phishing or malware infections [77, 222]. When users click on the spam content, they are often led to malicious websites that lure users to give away valuable information (*e.g.* passwords) or trigger drive-by download of malware exploits to compromise their computers [226].

Identity Theft. Today's online communities often put user information public by default (*e.g.*, full name, home address, birthday, location traces), which makes it possible for attackers to massively access the data via simple web crawling. Such information, once collected by attackers, can be used to facilitate more damaging attacks, from by-passing user's "security questions" in online banking [28], to manifesting highly customized spear phishing emails [32, 29]. As of 2015, the FTC received over 490,000 consumer complaints about identity theft, representing a 47% increase over the prior year [31].

Sybils. For most attacks in online communities (*e.g.*, social spam, identity theft), attackers often need to first control a large number of user accounts to carry out the attacks. While some of these accounts are stolen from real users (*i.e.*, compromised accounts) via Botnet [6], many more are massively created fake identities (*i.e.*, Sybils). Sybils are not uncommon in today's online services. As of 2015, Facebook has conservatively estimated that there are at least 170 million Sybils on their service, counting for almost 10% of the Facebook population [27].

Behind the massive Sybils and their attack activities is the growing marketplace for creating fake accounts [207, 215]. Other than attacks described above, Sybils have been used to create fake impressions or perceived popularity in social media [10]. For example, during the last US presidential election (2012), both candidates Mitt Romney and Barack Obama were spot to use fake Twitter followers to boost their perceived popularity for their campaigns [15, 14].

Cyberbullying. Online communities are not only facing with challenges from external attackers but also need to handle abusive behaviors from real users internally. One of the emerging issue is cyberbullying in online social networks, particularly, as more and more under-aged users (teenagers) are now using the services. Bullying behaviors are commonly found in traditional social media like Twitter [11], and recently get aggravated in anonymous social networks like Yik Yak, where people can post nasty messages about others anonymously [23]. Many highschools have prohibited the usage of the service on campus [24], which however cannot stop the damage. San Clemente, California, a high school was even shut down for a day after an anonymous bomb threat was posted on Yik Yak [25].

Key Challenges. My work seeks to improve security and privacy in online communities. To do so, we need to first understand the key challenges we are facing with. The *first* challenges is to gain a deep understanding on the problems. Relying on anecdote or pre-defined assumptions about users or attackers often leads to impractical solutions. For instance, earlier Sybil detection systems [247, 246, 217, 221, 69, 58] all rely on the assumption that Sybils would have difficulty in making friends with real users, and thus have to befriend with each other to form tight-knit Sybil communities. However, recent measurements show the opposite as real-world Sybils can successfully befriend with real users and blend into the social graph [244], rendering the proposed systems ineffective.

The *second* challenge is that attacker behaviors are constantly changing. On one hand, attackers would launch new attacks when new vulnerabilities are identified. On the other hand, as services providers deploy more advanced security defense, attackers need to adapt in order to survive. This creates a highly adversarial environment. For any security defenses, the challenge is not only to accurately detect existing attacks, but to identify previously unknown attacks and stay robust against adversarial adaptions of attackers.

Third, online communities not only face with external attacks but also internal abusive behaviors of real users. Being able to understand real user behaviors and activity patterns is critical to mitigate abuse. Today's online communities are huge in terms of the user population (*e.g.*, in millions or even billions). For such a large population, human behaviors can be highly diverse and often unpredictable, making this problem highly challenging.

Overview of My Work. In this dissertation, we seek to address the above challenges with a data-driven approach. First, we want to gain a deep understanding on behaviors of both real users and attackers in various online communities based measurements. We believe only by collecting and analyzing real-world data can we truly understand the security and privacy challenges online communities are facing with. Second, based on our measurement results, we then build data-driven systems for modeling user behaviors and detecting malicious attacks. Finally, we again use real-world data to evaluate the system performance and proactively test prototypes in realistic settings.

We execute the above methodology in 7 highly related projects and collectively present the results in this dissertation. We start with three measurement studies to discuss the security and privacy challenges faced by online communities with focus on quality of user content [223], anonymity and abuse [229], and location and user mobility [228]. Then in the second chapter, we dive into the problem of social spam campaigns driven by real Internet users [230] and explore possible defense techniques [229]. Finally we build data-driven solutions to detect previous unknown attacks [224] and use unsupervised models to capture and identity different user behaviors in online communities [231]. In the following, we give a briefly summary for each of the chapters.

1.1 Measurements of Online Communities

In this chapter, we provide contexts for today's online communities with focus on their user activities and the security and privacy challenges they are facing with. Based on large-scale datasets collected from real-world online services, we empirically examine how users behave and how they interact with each other at both global network level and individual user level. In addition, we take this chance to explore key questions regarding security and privacy. For example, how can online services with billions of users effectively maintain high-quality user generated content? How to preserve user privacy (anonymity) while maintaining accountability to constrain abusive behaviors? With the wide adoption of mobile devices, what are the emerging challenges in preserving user security and privacy while helping users to interact with the physical world efficiently?

Quality of User Content. First, we focus on Quora, one of the most successful social question and answering (Q&A) networks, to understand its key mechanisms to maintain high-quality user generated content. Recently, a number of question and answer (Q&A) sites have successfully built large growing knowledge repositories, each driven by a wide range of questions and answers from its users community. While sites like Yahoo Answers have stalled and begun to shrink, one site still going strong is Quora, a rapidly growing service that augments a regular Q&A system with social links between users. Despite its success, however, little is known about what drives Quora's growth, and how it continues to connect visitors and experts to the right questions as it grows. We present results of a detailed analysis of Quora using measurements. In this thesis, we shed light on the impact of three different connection networks (or graphs) inside Quora, a graph connecting topics to users, a social graph connecting users, and a graph connecting related questions. Our results show that heterogeneity in the user and question graphs are significant contributors to the quality of Quora's knowledge base. One drives the attention and activity of users, and the other directs them to a small set of popular and interesting questions.

Anonymity and Abuse. Second, we measure a popular anonymous social network Whisper to examine the trade-offs of anonymity of users and the accountability of users' behaviors. Social interactions and interpersonal communication has undergone significant changes in recent years. Increasing awareness of privacy issues and events such as the Snowden disclosures have led to the rapid growth of a new generation of anonymous social networks and messaging applications. By removing traditional concepts of strong identities and social links, these services encourage communication between strangers, and allow users to express themselves without fear of bullying or retaliation. Despite millions of users and billions of monthly page views, there is little empirical analysis of how services like *Whisper* have changed the shape and content of social interactions. In this thesis, we present results of the first large-scale empirical study of an anonymous social network, using a complete 3-month trace of the Whisper network covering 24 million whispers written by more than 1 million unique users. We seek to understand how anonymity and the lack of social links affect user behavior. We analyze Whisper from a number of perspectives, including the structure of user interactions in the absence of persistent social links, user engagement and network stickiness over time, and content moderation in a network with minimal user accountability. Finally, we identify and test an attack that exposes Whisper users to detailed location tracking.

Mobility and User Locations. Finally, we analyze Waze as an example of location-based crowdsourcing services to understand how manipulating the online information can impact the physical world. Waze is a mobile app of 50 millions users that provides timely updates on traffic, congestion, accidents and points of interest. In this thesis, we demonstrate how lack of strong location authentication allows creation of software-based *Sybil devices* that expose crowdsourced map systems to a variety of security and privacy attacks. Our experiments show that a single Sybil device with limited resources can cause havoc on Waze, reporting false congestion and accidents and automatically rerouting user traffic. More importantly, we describe techniques to generate Sybil devices at scale, creating armies of virtual vehicles capable of remotely tracking precise movements for large user populations while avoiding detection. Finally, we discuss possible solutions to mitigate this threat.

1.2 Spam, Human Factors and Malicious Crowdsourcing

In the second chapter, we specifically focus on the generation and distribution of malicious content (*e.g.* spam) in online communities and practical defense techniques. While traditional spam attacks are mostly generated by automated software, more sophisticated attackers today start to introduce "human intelligence" to their attacking process. Through extensive measurements, we find strong evidence on the rising of *malicious crowdsourcing* services where a large number of real users are hired for pennies to perform malicious activities, such as writing fake product reviews, creating fake social network accounts, and spreading rumors on social media. The result is a large volume of malicious and misleading content flooding into today's online communities. Malicious crowdsourcing poses a significant challenge to existing security systems (*e.g.*, CAPTCHA), which are initially designed to detect attacks from automated software, but become ineffective to real users. In this thesis, we describe our efforts in understanding and defending against malicious crowdsourcing (or *crowdturfing*).

Crowdturfing Measurements. Through measurements, we have found that malicious crowdsourcing systems are rapidly growing in both user base and total revenue. We use detailed crawls to extract data about the size and operational structure of these crowdturfing systems. We analyze details of campaigns offered and performed in these sites, and evaluate their end-to-end effectiveness by running active, benign campaigns of our own. Finally, we study and compare the source of workers on crowdturfing sites in different countries. Our results suggest that campaigns on these systems are highly effective at reaching users, and their continuing growth poses a concrete threat to online communities both in the US and elsewhere.

Defense and Adversarial Attacks. We then explore practical defense against crowdturfing activities. 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. To begin with, we examine the possibility to build machine learning classifiers to detect workers who perform crowdturfing tasks (on Weibo, Chinese Twitter). We show that traditional ML techniques are accurate (95%–99%).

However, ML models are 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 our context, we empirically evaluate the "robustness" of our classifier against a series of adversarial attacks using ground-truth data, including simple *evasion attacks* (workers modify their behavior) and powerful *poisoning attacks* (where administrators tamper with the training set). We find all of the tested classifiers are vulnerable to at least one of these adversarial attacks. 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.

1.3 User Behavior Modeling for Security Defense

In the final chapter, we develop data-driven systems to capture and understand (malicious) user behaviors, as a practical solution to combat fake identities and abuse in online communities. We build a novel user behavior model based on clickstream traces, *i.e.*, server-side sequences of click events generated by users when they are using the online service. The core of our proposal is the clickstream similarity graph, which uses similarity distance between pairs of clickstreams to capture user similarity. The result produces clusters that capture users with similar behavioral patterns. Based on this clickstream model, we develop two practical systems: The *first* system is a *semi-supervised* system to detect malicious user accounts (Sybils). We validate the system using ground-truth traces of 16,000 real and Sybil users from Renren, a large Chinese social network with 220M users. We demonstrate that our system achieves high detection accuracy with a minimal requirements of ground-truth inputs. The *second* system is an *unsupervised* system to capture more fine-grained user behavior. Instead of simply performing binary classification on users (either malicious or benign), this model identifies natural clusters of different user behaviors, and automatically extracts key features to interpret the captured behaviors. Applying this system to Renren and another real-world online social network Whisper (100K users), we help service providers to identify unexpected user behaviors (malicious accounts in Renren, hostile chatters in Whisper) and even predict users' future actions (dormant users in Whisper).

Both systems have received positive feedback from our industrial collaborators including Renren, LinkedIn and Whisper, after testing our prototypes on their internal clickstream data. Following positive results, these companies have expressed strong interest in further experimentation and possible internal deployment.

1.4 Contributions

This dissertation makes two high-level contributions to improving the security and privacy in large online communities. First, my work demonstrates the need for conducting real-world empirical measurements to understand the fundamental problems before designing any security mechanisms. Our measurements have repeatedly led to findings that contradict common assumptions made by either industrial service providers or prior research work. We have demonstrated the ineffectiveness of location fuzzing in anonymous social networks, and the attacking impact of Sybil devices due to the lack of reliable location authentications. In addition, we are the first to systemically study the use of “human intelligence” in social spam, and demonstrate their effectiveness in circumventing existing security mechanisms which assume attackers are automated software.

Second, we have designed, implemented and evaluated practical security solutions for large online communities. Whether it is the machine learning classifiers to detect crowdturfing workers or the clickstream-based graph model to detect Sybil accounts, we use real-world traces to drive the system design with focus on practicality. As a result, many of our system prototypes have been (or in the process to be) adopted by our industry partners including LinkedIn, Renren, Whisper and Microsoft. This helps to transfer our research efforts into real-world impact by providing protections for millions of Internet users.

Moving forward, as online communities continue to grow, they are likely to receive even more attentions from malicious parties. In particular, with the fast development of personal Internet devices ranging from wearables, to Internet-connected vehicles, to smart houses, we are very likely to have more online communities formed in the near future where new types of user interaction and data exchange occur. With this process continuing to shape the surface of security and privacy, researchers should always expect new attacking strategies from attackers and their evasions against our defense. This dissertation demonstrates a first step along this path: by combining inter-disciplinary tools such as measurements, graph analytics, machine learning and human factor study, we bring clarity to the problems and develop practical systems to better prepare users and online services to step up the cybersecurity game.

Chapter 2

Understanding Online Communities via Measurements

In this chapter, we seek to provide contexts for today's online communities, their users' activities and most importantly the security and privacy challenges they are facing with. Our key methodology is to conduct *measurements*. By collecting and analyzing large-scale datasets from real-world online services, we empirically examine user activities in those systems and answer the following key questions. First, how can online services with billions of users effectively maintain high-quality user generated content? Second, how to effectively preserve user privacy (anonymity) while maintaining accountability to constrain abusive behaviors? Third, with the widely-adopted mobile devices, what are the emerging challenges in preserving user security and privacy while helping users to interact with the physical world more efficiently?

In the following, we present three empirical measurement case studies on selected services to discuss the above questions that broadly cover quality of user content, anonymity and abuse, and location and user mobility. Through the three case studies, our goal is to provide a deeper understanding on the problem space and seek for possible directions of solutions.

2.1 Quality of User Content

2.1.1 Introduction

The Internet is a maelstrom of information, most of it real, and much of it false. Efforts such as Wikipedia have shown that collectively, Internet users possess much knowledge on a wide range of subjects, knowledge that can be collated and curated to form valuable information repositories. In the last few years, community question-and-answer (Q&A) sites have provided a new way for users to crowdsource the search for specific detailed information, much of which involves getting first-hand answers of specific questions from domain experts.

While these sites have exploded in popularity, their growth has come at a cost. For example, the first and still largest of these sites, Yahoo Answers, is showing clear signs of stalling user growth and stagnation, with traffic dropping 23% in a span of four months in 2011 [148]. In addition, the Google Answers service launched in 2001 was already shut down by 2006. Why is this the case? One of the prevailing opinions is that as sites grow, a vast number of low-value questions overwhelm the system and make it extremely difficult for users to find useful or interesting content. For example, ridiculous questions and answers are so prevalent on Yahoo Answers that a quick Google search for “Yahoo Answers Fail” turns up more than 8 million results, most of which are sites or blogs dedicated to documenting them.

Bucking the trend thus far is Quora, an innovative Q&A site with a rapidly growing user community that differs from its competitors by integrating a social network into its basic structure. Various estimates of user growth include numbers such as 150% growth in one month, and nearly 900% growth in one year [148]. Despite its short history (Quora exited beta status in January 2010), Quora seems to have achieved where its competitors have failed, *i.e.* successfully drawing the participation of both a rapidly growing user population and specific domain experts that generate invaluable content in response to questions. For example, founders of Instagram and Yelp answered questions about their companies, Stephen Fry and Ashton Kutcher answered questions about actors, and domain-specific answers come from experts such as Navy Seals sharpshooters and San Quentin inmates.

So how does Quora succeed in directing the attention of its users to the appropriate content, either to questions they are uniquely qualified to answer, or to entertaining or informative answers of interest? This is a difficult question to answer, given Quora's own lack of transparency on its inner workings. While it is public knowledge that Quora differs from its competitors in its use of social networks and real identities, few additional details or quantitative measures are known about its operations. A simple search on Quora about how it works produces numerous unanswered questions about Quora's size, mechanisms, algorithms, and user behavior.

In the first part of this chapter, we perform a detailed measurement study of Quora, and use our analyses to shed light on how its internal structures contribute to its success. To highlight key results, we use comparisons against Stack Overflow, a popular Q&A site without an integrated social network. We seek to answer several key questions:

- What role do traditional question topics play in focusing user attention? How much do followers of a topic contribute to answering its questions?
- What impact do super users have on specific patterns of user activity? Can they generate and focus user attention on individual questions, thus setting them apart from questions on related topics?
- Given the rapid growth of questions on question-and-answer sites, how does Quora help users find the most interesting and valuable questions and avoid spammy or low-value questions? What role do the "related questions" feature play?

2.1.2 Background: Quora

Quora is a question and answer site with a fully integrated social network connecting its users. In this section, we introduce Quora, using Stack Overflow as a basis for comparison. We then give details on the key Quora graph structures that connect different components together. Specifically, we describe three types of graphs in Quora: a social graph connecting users, a user-topic following graph and a related question graph.

2.1.2.1 Quora and Stack Overflow

Quora. Quora is a question and answer site where users can ask and answer questions and comment on or vote for existing answers. Unlike other Q&A sites where all users exist in a global search space, Quora allows users to follow each other to form a social network. Social connections in Quora are directional like Twitter. A user A can follow user B without explicit permission, and B's actions (new questions, answers, comments and topics) will appear in A's activity stream. We say A is B's *follower* and B is A's *followee*. In addition, users can follow *topics* they are interested in, and receive updates on questions and answers under this topic.

Each Quora user has a *profile* that displays her bio information, previous questions and answers, followed topics, and social connections (followers and followees). Each user has a "Top Stories" page, which displays updates on recent activities and participated questions of their friends (followees), as well as recent questions under the topic they followed. A small subset of registered users are chosen by Quora to be *reviewers* and *admins*, and have the power to flag or remove low quality answers and questions.

Finally, each Quora question has its own page, which includes a list of its answers and a list of related questions. Users can add new answers, and comment, edit and vote on existing answers.

Stack Overflow. Stack Overflow is another successful Q&A site started in 2008. Stack Overflow differs from Quora in two main aspects. First, while Quora covers a broad range of general topics, Stack Overflow focuses specifically on computer programming questions. Second, users in Stack Overflow are fully independent without social connections.

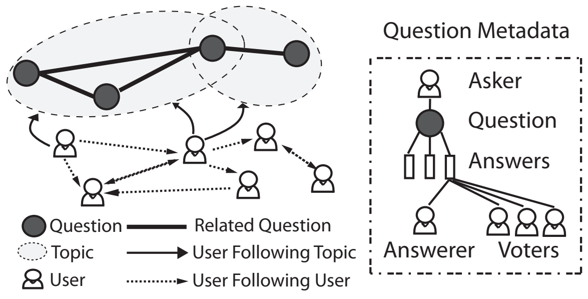


Figure 2.1: Structure of questions, topics and users in Quora.

2.1.2.2 Graph Structures In Quora

The internal structure of question-and-answer sites are often a complex mix of questions, answers, question topics, and users. We summarize the relationships between different entities in Figure 2.1. Users can follow individual topics and other users for news and events; questions are connected to other "related" questions, and each question can be tagged with multiple topics. Finally, for each question in the system, there is a user who asked that question (the *asker*), users who answered that question (*answerers*), and users who voted on an answer (*voters*).

Quora's internal structure is dominated by three graphs that act as channels that guide user interest and deliver information to users.

1. *User-Topic Graph:* Quora users follow different topics, and receive updates about questions under topics they follow.
2. *Social Graph:* Quora users follow each other to form a Twitter-like social graph. Users receive newsfeed about questions their friends participated in.
3. *Question Graph:* Each question has a list of related questions used by users to browse related questions. The "related" relationship is considered symmetric.

We believe these three graphs are largely responsible for guiding the attention of Quora users. In the following, we perform detailed analysis on these graphs to understand how they

Website	Data Since	Total Questions	Total Topics	Total Users	Total Answers
Stack Overflow	Jul. 2008	3.45M	22K	1.3M	6.86M
Quora	Oct. 2009	437K	56K	264K	979K

Table 2.1: Data Summary.

impact user activities, especially how they help users separate a small subset of interesting questions from the larger number of less interesting questions/answers.

2.1.3 Dataset and Preliminary Results

Before diving into main analytical results of our work, we begin in this section by first describing our data gathering methodology and presenting some preliminary results. Here we describe the properties and limitations of our Quora and Stack Overflow datasets. We also analyze some high level metrics of the Quora data, while using Stack Overflow as a baseline for comparison.

2.1.3.1 Data Collection

Our analysis relies on two key datasets. A publicly available dataset periodically released by Stack Overflow, and a dataset crawled from Quora that contains multiple groups of data on users, questions, topics and votes. We describe details below. The basic statistics of both datasets are shown in Table 2.1.

Stack Overflow. Stack Overflow periodically releases all of their data to the public. Our site trace was released in August 2012, and covers all activity on Stack Overflow between July 2008 and July 2012.

Quora. We gathered our Quora dataset through web-based crawls between August and early September 2012. We tried to follow crawler-etiquette defined in Quora's `robots.txt`. Limited portions of data were embedded in Ajax calls. We used FireWatir, an open-source

Ruby library, to control a FireFox browser object, simulating clicking and scrolling operations to load the full page. We limited these crawls to 10 requests/second to minimize impact on Quora.

Since Quora has no predefined topic structures for its questions (questions can have one or more arbitrary topic "labels"), getting the full set of all questions is difficult. We followed the advice from a Quora data scientist [16] and start our question crawls using 120 randomly selected questions roughly evenly distributed over 19 of the most popular question topics. The crawls follow a BFS pattern through the related questions links for each question. In total, we obtained 437,000+ unique questions. Each question page contains the topics associated to the question, a complete list of answers, and the answerers and voters on each answer. As shown in Table 2.1, this question-based crawl produced 56,000+ unique topics, 979,000+ answers, and 264,000+ unique users who either asked or answered a question, or voted on an answer.

Our biggest challenge is trying to understand how much of the Quora dataset we were able to gather. The simple answer is we don't know, since there are no official quantitative measures about Quora available. But we found a post by a Quora reviewer [13] that hinted the question ID (or *qid*) in Quora is sequentially assigned. Thus we can infer the total number of questions by inspecting the qid of the newly added questions. To validate this statement, we performed several small experiments where we added small bursts of new (meaningful) questions to Quora. Each burst contains 10 new questions sent seconds apart, and consistently produced 10 sequential qid's. We separated experiments by at least 30 minutes, and observed increments to the qid consistent with the expected number of new questions in the gap between experiments. Finally, we plotted qid values for all questions found by our crawl and correlated them with the estimated date of question creation. The result, discussed below, provides further support that this qid can be used as an estimate of total questions in the system. The largest qid from our crawled questions is 761030, leading us to estimate that Quora had roughly 760K questions at the time of our crawl, and our crawl covered roughly 58% of all questions. Note that not all questions remain on the site, as Quora actively deletes spam and redundant questions [18]. This estimate might provide an upper bound of actual number of questions, and our coverage of 58% would be a lower bound.

We also crawled the user profiles for users extracted from the crawled questions. Each user profile contains 6 parts: the list of the user's followers, list of users they follow (followees), their previous answers, their previous questions, their followed topics and boards. Out of the 264K extracted users, we found that roughly 5000 (1.9%) profiles were no longer available, likely deleted either by Quora or the user.

***Qid* Over Time.** Assuming we are correct about the use of qid, we can plot an estimate of the growth of Quora (and Stack Overflow), by plotting qid against time. Since Quora does not show when a question is posted, we estimate the posting time by the timestamp of its earliest answer. For open questions with no answer, we infer the question posting time based on the latest activity timestamp on the question page. Since reading the question does not update this "latest activity" timestamp, this timestamp can estimate posting time for unanswered questions. We estimate the total number of questions in Quora for each month by looking at the largest *qid* of questions posted in that month. For Stack Overflow, we use the timestamp for questions creation in the data trace.

We see in Figure 2.2 that Stack Overflow is an older site with more questions than Quora. We plot two lines for Quora, a black dashed line for the total number of questions estimated by *qid*, and the blue dashed line is the number of questions we crawled from each month. Both lines increase smoothly without gaps, suggesting that Quora did not reset *qid* in the past and the questions we crawled are not biased to a certain time period. Our estimated number of questions in Quora for June 2012 is 700K, which is consistent with previously reported estimates [149]. As Quora continues to grow, it is clear that helping users easily identify and find the most meaningful and valuable questions and answers is a growing challenge.

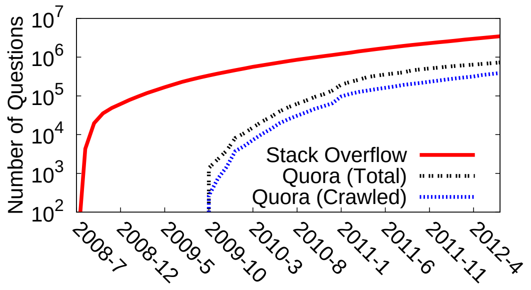


Figure 2.2: Questions growth.

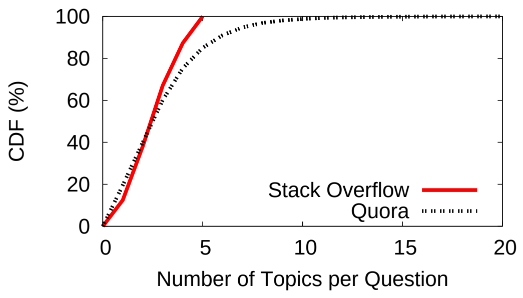


Figure 2.3: # of Topics per question.

Topic in Quora	# of Questions	Topic in Stack Overflow	# of Questions
Startups	16.3K	C#	333K
Survey Questions	10.3K	Java	277K
Movies	9.7K	PHP	257K
Medicine / Healthcare	9.3K	Javascript	242K
Food	8.7K	Android	211K
Facebook	7.4K	jquery	207K
Music	5.5K	iPhone	143K
Google	5.4K	C++	139K
Psychology	5.2K	ASP.net	132K
Startup Advice	5.2K	.net	125K

Table 2.2: Top 10 topics based on number of questions.

2.1.3.2 Initial Analysis

Topics. Quora is a general Q&A site with a very broad range of topics. We observed 56K topics in our dataset, which is twice more than that of Stack Overflow, even though Quora is smaller by question count. Table 2.2 lists the top 10 topics with most number of questions in each site. In Quora, the top 10 includes topics in various areas including technology, food, entertainment, health, etc. "Startups" is the most popular one which takes 3.7% of the questions. While all topics in Stack Overflow are different, they are all related to programming. The most popular topic is "C#," which represents roughly 10% of all questions.

Questions and Answers. In both systems, one question can have multiple topics. Figure 2.3 shows the number of topics per question. Stack Overflow requires a minimum of 1 topic and

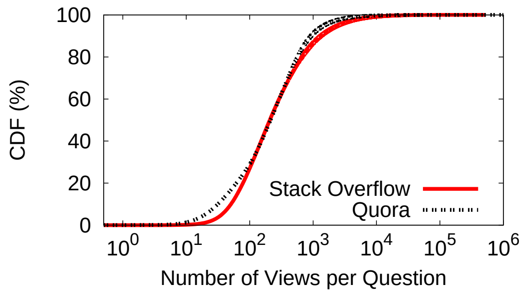


Figure 2.4: # of User views per question.

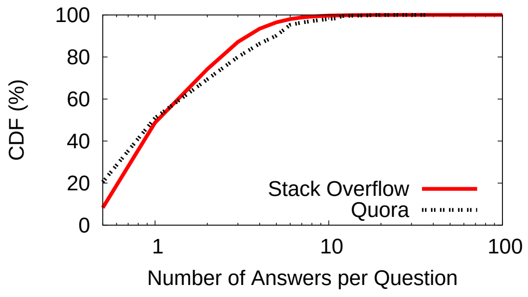


Figure 2.5: # of Answers per question.

a maximum of 5 topics per question, and the results are evenly distributed between 1 and 5. Although Quora does not have such requirements, a majority (85%) of questions have no more than 5 topics. Very few (<1%) of questions end up with more than 10 topics, which might be an attempt to draw more attention to the question.

Next, we plot the distribution of views and answers per question in Figure 2.4 and Figure 2.5. We are surprised to find that the curves from Stack Overflow and Quora are nearly identical. Although 20% of questions in Quora remain unanswered (10% for Stack Overflow), almost all questions got at least 1 user view. In addition, 99% of questions end up with less than 10 answers, and 20% of all Quora questions managed to collect $\geq$4 answers. We use this as a minimum threshold for our later analyses on social factors on system performance.

Votes. In terms of votes, both Quora and Stack Overflow allow users to upvote and downvote answers. Quora makes visible the list of upvoters, but hides downvoters. Downvotes are processed and only contribute to determining the order answers appear in. Thus in our analysis of Quora, we only refer to upvotes and disregard downvotes. In contrast, Stack Overflow anonymizes all voters and only displays the accumulated number of votes, which can be negative if an answer is poorly received.

Next, we look at how votes impact the order that answers are displayed. Quora uses a proprietary algorithm [9] to rank the answers, where best answers show on the top of the page. In Stack Overflow, the question asker can *accept* one of the answers as the best answer. First,

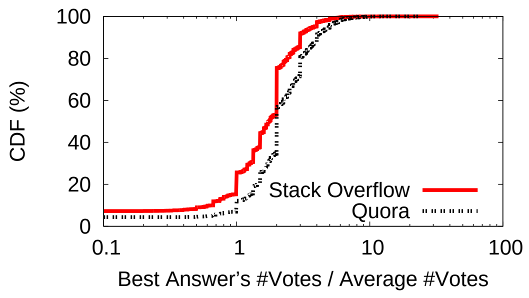


Figure 2.6: Votes for the best answer vs. the average.

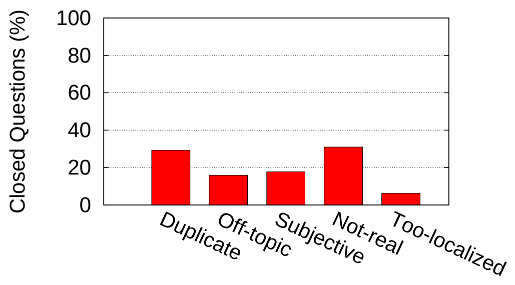


Figure 2.7: Reasons for deleting questions in Stack Overflow.

we examine how well votes work to identify the "best answer." We select questions with at least 2 answers, 180K or 40% of all questions in Quora and 1.76M or 51% in Stack Overflow. Figure 2.6 plots the ratio of the best answer' votes over the average votes per answer under this question. We call this as "best answer vote ratio." Overall, vote count was very effective at identifying the best answers, and the differences between the two sites might be explained by the more concrete (right or wrong) nature of Stack Overflow's questions compared to general questions on Quora. Surprisingly, some of the best answers have less votes than the average answer. 5% of Quora questions ranked answers with fewer upvotes on top, likely due to other features used by Quora's ranking algorithm such as answerer reputation or downvotes. On Stack Overflow, 7% of the answers chosen by the asker had lower votes than average.

Content Moderation. Finally, we note that both sites use crowdsourcing to moderate user-generated content. Stack Overflow has administrators who actively flag unqualified questions and close them [17]. Roughly 3% of all questions in Stack Overflow have been closed, and Figure 2.7 shows the reasons why they were closed. The top two reasons were "not-real," *i.e.* ambiguous, vague, incomplete, overly broad or rhetorical, and redundant questions. In contrast, Quora relies on a total of 43 admins and 140 reviewers chosen from the user population to flag low quality answers and redundant questions [19, 20]. The number of flagged or removed answers and questions is unknown. While it is unclear whether these reviewers are responsible for keeping Quora largely free of fake or scripted accounts (Sybils) [230, 244], recent work has shown that human reviewers can be extremely effective at detecting fake or forged content [225].

Summary. Despite their different topics of interest, Quora and Stack Overflow share many similarities in distribution of content and activity. A key observation is that given the broad and growing number of topics in Quora, identifying the most interesting and useful content, *i.e.* separating the wheat from the chaff, is a very difficult problem. Without built-in mechanisms to lead users to useful content, the service will overwhelm users with the sheer volume of its content, much like the Internet itself. This is the focus of the rest of analysis, where we will study different Quora mechanisms to understand which, if any, can keep the site useful by consistently guiding users to valuable information.

2.1.4 The User-Topic Graph

Quora allows users to track specific fields by following the corresponding topics, such as “Startups,” “Facebook,” and “Technology.” This also directly connects users to questions (and associated answers). A question, once created or updated under a topic, will be pushed to the newsfeeds of users who follow the topic. In this section, we model the interaction between Quora users and topics using a *user-topic graph*, and examine the impact of such interactions on question answering and viewing activities. Specifically, we seek to understand whether there is a direct correlation between followers of a topic and views and answers to questions, *i.e.* do highly-followed topics draw a large number of views and answers to their questions?

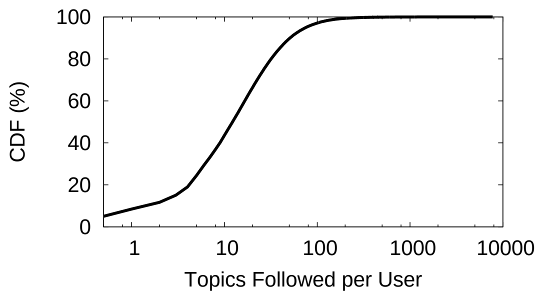


Figure 2.8: Topics followed per user.

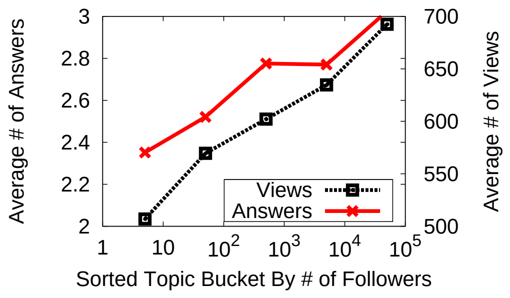


Figure 2.9: Average views and answers for questions under sorted topic buckets.

2.1.4.1 High-level Statistics

We first examine the number of topics followed by each user[1]. Figure 2.8 shows the cumulative distribution of the number of topics followed per user. We make three key observations. First, the large majority (95%) of users have followed at least 1 topic. This is because Quora recommends topics during the sign-up process. Second, Quora users each tend to follow a moderate number of topics, *e.g.* more than 50% of users followed at least 10 topics, but 97% of users followed no more than 100 topics. Finally, a very small portion of users (27 or 0.01%) followed more than 1000 topics. We manually checked these users and found that they were legitimate accounts, and come from various backgrounds such as CEOs, co-founders, bloggers, students, and were all very active Quora users.

2.1.4.2 Impact on Question-related Activities

We now examine whether user interest towards certain topics translates into higher level of activities on questions related to those topics. We examine the correlation between the number of views or answers per question, and the number of followers of each topic. Since the number of topics is large (35K), we bucketize topics based on the number of followers in a log scale. For example, topics with number of followers in the range $[1, 10]$ are in one bucket, and topics

[1]The user-topic interaction is one-way where users can follow multiple topics, but the relation is asymmetric, *i.e.* topics do not follow users.

with number of followers within $[10, 100]$ are in a second bucket. We have a total of 5 buckets. In each bucket, we compute the number of views (answers) per question, averaged over the topics and their questions.

Figure 2.9 shows the correlation results for both question views and answers. We observe a strong correlation: questions under topics with more followers tend to have a higher number of average page views and answers. This is intuitive: when a user follows a topic, all questions under the topic and their updates show up on the user's newsfeed, thus encouraging page views and answers.

We verify this intuition by examining for each question the percentage of answers that came from followers of the question's topic(s). Unfortunately we could not do the same for question page views, because Quora only reveals the identity of users who answer questions, but not those who browse each question. We focus on questions with some minimum number of user interactions ($\geq$4 answers), which filters out all but 87K (20%) questions from our dataset. Figure 2.10 plots the cumulative distribution of the portion of answers contributed by topic followers. It is very close to a uniform distribution with mean of 50%, except for roughly 13% of questions, for which none of the answers were produced by followers of the question's topic(s). At a high level, this suggests that topics are effective ways of guiding users towards questions that are valuable and appealing to them.

Summary. The user-topic interaction has considerable impact on question answering activities in Quora. Not surprisingly, questions under well-followed topics generally draw more answers and views. Following the right topics can introduce users to valuable questions and answers, but is not the only way to access questions.

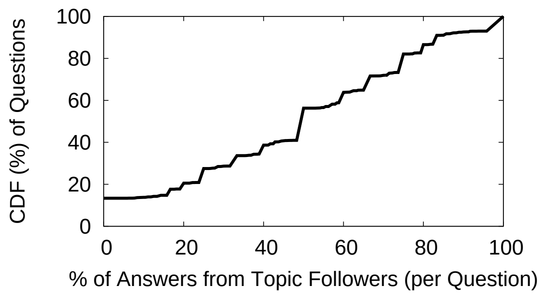


Figure 2.10: % of Answers added by the followers of the question's topics.

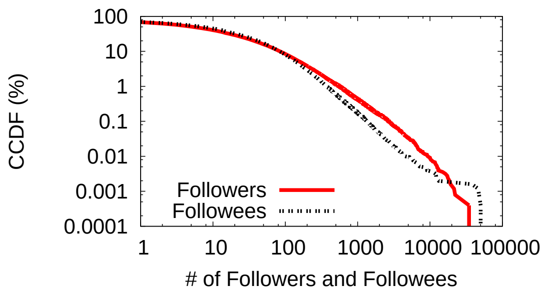


Figure 2.11: Degree distribution in social graph.

2.1.5 The Social Graph

In addition to following topics of interest, Quora users also follow each other to form a Twitter-like directed social graph. Questions that a user interact with are dissiminated to their followers in the form of events in their newsfeed. Therefore, social relationships clearly affect Q&A activities, and serve as a mechanism to lead users to valuable information.

In this section, we analyze the Quora social graph to understand the interplay between user social ties and Q&A activities. Specifically, we seek to answer three key questions. First, what triggers Quora users to form social ties? Second, does the presence of popular users correlate with high quality questions or answers? That is, do questions raised by "super-users" with many followers receive more and/or better answers from her followers? Finally, do strong social ties contribute to higher ratings on answers to questions? In other words, do questions answered by super-users get more votes because of the sheer number of their followers?

2.1.5.1 Social Ties

We begin by examining the follower and followee statistics of Quora users. Figure 2.11 plots the complementary cumulative distribution function (CCDF) for both the incoming degree (follower) and outgoing degree (followee). As expected, the degree distribution follows the power-law distribution [45]. Specifically, 23% of users have no followers and 23% do not follow anyone. The vast majority of users (99.6%) have less than 1000 followers, while 23 users have more than 10,000 followers. The exponential fitting parameter α for the incoming degree distribution is 2.49 (with fitting error 0.01). This is very close to that of Twitter (α=2.28), but higher than that of Facebook and Orkut (α=1.5) [236, 151].

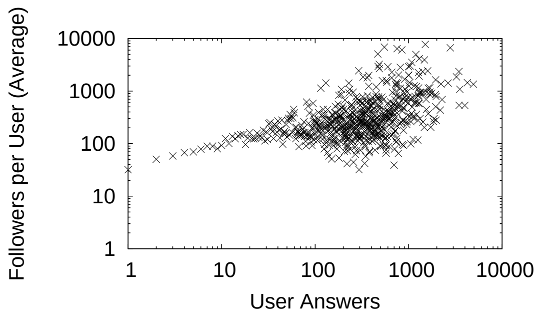
In our data set, 44,091 (17%) of all users have neither followers nor followees. For the rest, 6% of users have no followers, and 7% do not follow anyone, representing the two extremes in the FFRatio distribution. Overall, more than half (58%) of all users have more followees than followers. A very small portion (0.1%) have 100 times more followers than followees. Not surprisingly, these are mostly celebrities, *e.g.* editors, actors and CEOs.

Triggers of Social Ties. To understand how Quora's social network functions, a basic question of interest is how users choose their followees. According to a recent survey of Quora users [174], they tend to follow users who they consider interesting and knowledgeable. Thus our hypothesis is that, outside of the small portion of celebrities who get followers just by their mere presence, the majority of Quora users attract followers by contributing a large number of high-quality answers.

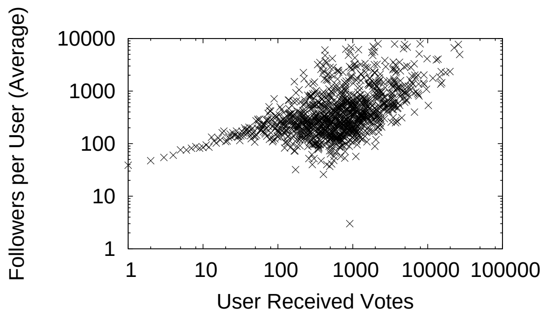
To validate our hypothesis, we examine the correlation between a user's follower count and the quantity and quality of her answers to questions. We approximate the quality of an answer by the number of votes received. We put users with the same number of answers (votes) into a group and compute the average number of followers per user for each group. Figure 2.12a plots the correlation results, which confirm our hypothesis. The correlation is particularly strong for users with less than 100 followers, which account for 91% of the users in our dataset.

2.1.5.2 Impact on Question Answering

Quora is unique because it integrates an effective social network (shown above) into a tradition Q&A site. Thus it is important to understand how social ties affect Q&A activities.



(a) Answers



(b) Received Votes

Figure 2.12: Correlation between user answers (received votes) and followers per user.

Specifically, we explore whether super users (or users with many followers) draw more and better answers from their followers. To answer this question, we first examine for each question the number of answers, and the portion of answers coming from the asker's followers. We then measure the quality of answers based on votes and explore whether followers provide better answers. We define "Super User" as top 5% of all users by followers. In our dataset, we have 12K super users, each with more 160 followers.

For questions in our dataset, the asker is not shown on the question page. Instead, we match the originator of the question (the "asker") to each question based on user profiles. Each user's profile page contains a list of user's previously asked questions. Using this list, we managed to find the askers for 285K (65%) questions in our question dataset. Since our analysis targets user social activities in the question thread, we do not consider open questions and questions that have not gained enough answers. We only consider questions with known askers and at least 4 answers, which still leaves a large number of questions (59K) for our analysis.

Number of Answers. In Figure 2.5 we have plotted the distribution of the number of answers received per question across all the questions. We repeat this analysis for both questions raised by super users and non-super users (regardless of the number of answers received), and found that they follow the same distribution (figure omitted due to the space limitations). This shows that users do not get more answers for questions just by having more followers.

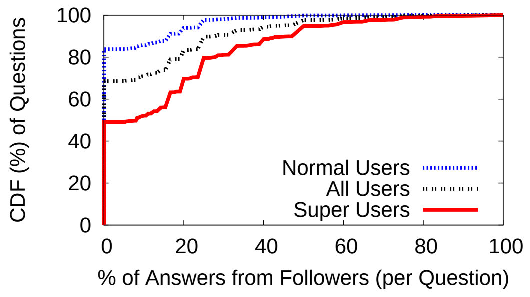


Figure 2.13: % of Answers written by asker's followers.

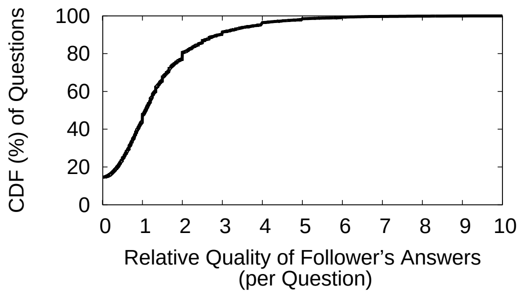


Figure 2.14: Relative answer quality ratio.

Answers by Followers. Next, we examine for each question the portion of answers contributed by the asker's followers. Figure 2.13 plots the cumulative distribution across all the questions (marked as "All"), across the questions raised by super users ("Super User"), and across the questions raised by non-super users ("Normal").

We make two key observations. First, a big portion of the questions (68% for "All") did not receive answers from the asker's followers. Even half of the questions raised by super users received no answers from their followers. This is likely because users who follow someone tend to seek her (helpful) answers to questions, rather than looking for questions to answer. This also implies that if we build a Q&A site *solely as a social network* that expects answers only from friends (followers), most questions will remain unanswered. Second, compared to normal users, super users do draw more answers from their followers, indicating a moderate level of social influence on question answers.

We also compare the effectiveness of drawing answers using social ties to that of drawing answers from following topics (discussed in Section 2.1.4), by comparing the results in Figure 2.13 and Figure 2.10. We see that in general, questions received more answers from users who follow the associated topic(s). But neither channel appears to be the primary way of attracting answers, and both channels appear to complement each other in this process.

Answer Quality. We now examine whether answers contributed by the asker's followers have better quality. Again we use the number of votes received to serve as an approximate measure of the quality of an answer. For each question thread, we first compute the average votes per answer for all the answers V_{all} and for all the answers contributed by the asker's followers $V_{follower}$. We define $R = \frac{V_{follower}}{V_{all}}$ as the relative quality of the followers' answers. Thus $R > 1$ indicates that the followers' answers are of higher quality in general.

Figure 2.14 shows the cumulative distribution of R, where for more than 50% of the questions, answers from the followers were of higher quality, and for 20% of the questions, answers from the followers got more than 2 times the votes than average. This result is consistent with a recent survey study [154] on Q&A behaviors in Facebook, which suggests that close friends have stronger motivation to contribute good answers.

2.1.5.3 Impact on Voting

Quora applies a voting system that leverages crowdsourced efforts to promote good answers. By positioning good answers at the top of the questions page, Quora allows users to focus on valuable content. However, the social interaction among Quora users could impact voting in various ways. The key concern is users who have many followers can get their followers to vote for their answers, thus gaining an "unfair advantage" over other users. In the following, we study this issue in detail by exploring two key questions. First, do user votes have a large impact on the ranking of answers in Quora? Second, do super users get more votes, and do these votes mainly come from their followers?

Votes and Ranking. Quora has indicated that the number of votes is the key metric to determine quality of answers [9]. In fact, our results in Figure 2.6 show that more than 96% of the best answers (ranked 1st by Quora) received more votes than average. Thus our goal is to explicitly examine how much the number of votes matters in Quora's ranking algorithm, and whether social connections give user advantage to gain more votes.

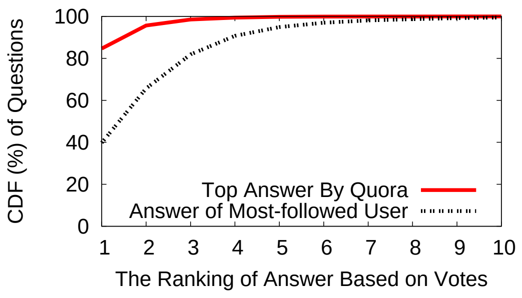


Figure 2.15: Ranking answers by author popularity vs top answer selected by Quora.

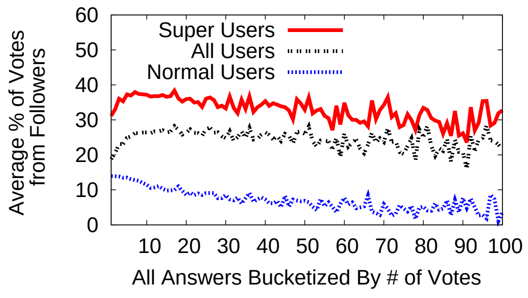


Figure 2.16: Votes from answerer's followers.

For each question thread, we start by ranking the answers by the number of votes received. Answers with the most votes are ranked first. We then take the best answer (ranked 1st) chosen by Quora's built-in algorithm and study their vote-based ranking. Figure 2.15 plots the cumulative distribution of these best answers' vote-based ranking. We see that for 85% of the questions, Quora's best answers also ranked the highest in votes, and for 96% of the questions, the best answers from Quora are among the top-2 most votes. This result confirms that the number of votes is the dominating feature for selecting best answers. The same result also implies that potential bias in the voting process could lead to unfair ranking of answers, which we study next.

Votes on Super Users. We repeat the above analysis on answers offered by super users (most followed users). Results in Figure 2.15 show that for 40% of questions, super users' answers received the highest votes, and for 60% of cases, their answers are among the top-2 most votes. This implies that regardless of the quality of their answers, super users can often get more votes over other users.

To better understand the bias, we examine whether a large portion of votes come from the answerer's followers. For this we gather answers to all the questions and group them by the number of votes received. For each group of answers, we compute the average percentage of votes from the answerer's followers. We also repeat the same process on answers offered by super users and those from non-super users. Figure 2.16 shows the average percentage of followers' votes across different answer groups. We cut the line at the points where the number of votes reaches 100, which covers 99.9% of all answers These results show that answers contributed by super users do receive a large portion of votes (30-40%) from their followers, which is significantly larger than normal users ($<$10%). This shows that users with more followers tend to get more votes from their followers, which could introduce potential unfairness in answer ranking. For example, an answer contributed by super users gets a much higher rank even though the true quality of the answer is not high.

Summary. In Quora, users who contributed more and good answers tend to have more followers. These well-connected users also gain advantage by having more friends (followers) to answer their questions and upvote for their answers.

2.1.6 The Related Questions Graph

One of Quora's core features is the ability to locate questions "related" to a given question. This effectively creates a *related question graph*, where nodes represent questions, and links represent a measure of similarity as determined by Quora. The related question graph provides an easy way for users to browse through Quora's repository of questions with similarity as a distance metric.

In this section, we extract the question graph from our dataset, and seek to determine if the structure of the graph plays a role in helping users to find top questions. Intuitively, a similarity-based question graph would produce large clusters of questions around popular topics, with less popular questions relegated to sparse regions of the graph. Thus users following related question links could encounter popular questions with a higher probability.

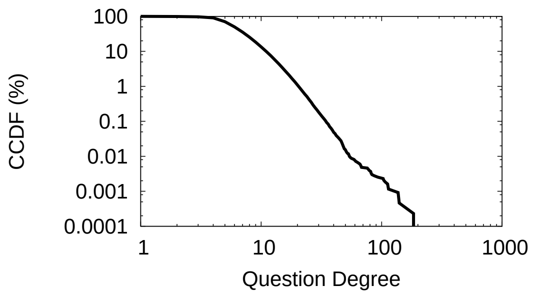


Figure 2.17: Node degrees in the related question graph.

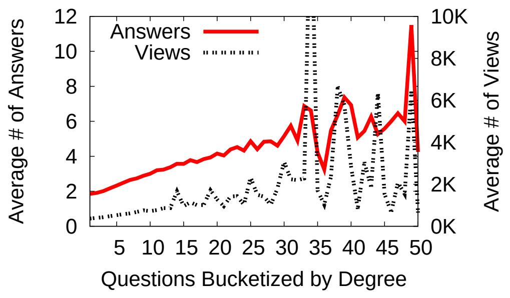


Figure 2.18: Question degree versus average views and answers per question.

2.1.6.1 Impact of Degree in the Question Graph

We build the question graph by crawling and extracting related questions links. By default, Quora lists a fixed number (5) of related questions on each question's main page. These are deemed by Quora to be the most related to the question on the current page. Since the "related" relationship is intuitively a bidirectional property, the question graph is a unweighted, bidirectional graph.

Our final question graph has a total of 437K nodes and 1.6M edges. We plot the distribution of question degree in Figure 2.17. Although each question only has at most 5 outgoing related questions, most questions have incoming connections from other questions, and thus have a total "related" degree greater than 5. However, there are 9% questions with degree less than 5. This is because some of their related questions were not crawled (questions deleted by Quora) and thus are not included as nodes. 99% of the questions have degree less than 50. The distribution shows a distinctive power-law shape, and when we fit the question degree CCDF to the power-law, we get an α value of 3.5 with fitting error 0.048.

Next, we examine the connectivity of the question graph. The question graph is dominated by a single large connected component that covers 98% (430K) of all questions. On closer inspection, we see that the remaining 2% of the questions are either newer questions whose related questions have not yet been computed, or they belong to esoteric topics with very few questions and low user interest.

Stability. One concern we had about the question graph is whether it is stable, *i.e.* does it change on a frequent basis as new questions are added to the system. We test the long-term stability of the related question graph by comparing the related question graph across two snapshots. The first snapshot was taken in our primary measurement period of August 2012. We also took another snapshot in October 2012 (two months after the first snapshot). When we compared the related question set for each question in the system, we found that 60% of all question had no changes in the time between our snapshots, and 30% of the questions have only one new entry (out of five) in its related question list. Thus we can assume that the related question list is relatively stable over moderate time periods, and our snapshots are a reasonable approximation for earlier versions of the question graph.

Question Degree vs. Attention. On each question page, users can browse a series of questions via the related question edges. This leads to the hypothesis that a question with higher question degree can receive more attention, *i.e.* more user views, and potentially more answers as a result.

We validate this hypothesis as follows. We first group all questions based on question degree in the related question graph. Then we compute the average number of answers and views for questions in each group. We plot the results in Figure 2.18. The dashed line represents the average number of user views across all questions with a given node degree, and the solid red line represents the average number of answers received by all questions with a given degree. There are clear trends in both cases. For questions with higher degree (they are listed as being related to more questions), they are accessible to users via a larger number of incoming links. Hence these high degree questions receive both more page views as well as more answers. The takeaway here is that questions with high degree in the question-relation graph correlates strongly to questions that receive more attention and answers from users.

2.1.6.2 Locating Similar Questions

In the question graph, questions on similar topics are clustered together, while irrelevant questions are likely to be "related" to popular questions. Thus they are likely positioned in sparser regions of the graph. In this subsection, we first leverage the graph structure to identify groups of similar questions. We then ask two key questions: do similar questions receive equal attention from users? If not, what are the potential mechanisms that drive users to certain questions while ignoring other similar questions?

Graph Clustering. We first locate similar questions using the question graph. More specifically, we want to generate question clusters where questions within the cluster are more tightly connected than those outside the cluster. This is a simple definition easily characterized by modularity.

We formalize this problem as a graph partition problem, and use the popular graph partitioning tool METIS to perform a multilevel k-way partitioning [121] on our question graph. In this case, we predefine K as the number of clusters we want to generate. We run the graph partitioning algorithm, with K equal to 100, 1000, 10,000 and 100,000. When K is too big, we end up with many small clusters after cutting many edges. On the other hand, when K is too small, we get a small number of big clusters which take in many questions under related topics, but are not truly similar. Since there is no good way to get the ground-truth assessment on how "similar" the questions are, we randomly sample 10 clusters from each run with different K values, and manually inspect questions within each cluster. We find that the best match between semantic clusters and automatically detected clusters occurs when $K =$10,000.

So we partition the graph into 10,000 clusters of similar sizes. Table 2.3 shows an example of one generated cluster. This cluster contains 43 questions, and all questions are related to "Quora." We also extract the topics of the questions in the cluster and rank the topics based on how many questions they are associated with. The top 3 topics of the cluster are listed in the

ID	Question Title
459576	What percentage of questions on Quora have no answers?
370857	Can I search Quora only for questions that have been answered?
45022	How many questions have been answered on Quora?
20195	What percentage of Quora questions receive at least one answer?
17363	What percentage of questions on Quora are answerable?
13323	How many questions are on Quora, answered or not?
...	...
Top Topics	Quora, Quora-Usage-Data-and-Analysis, Quora-product

Table 2.3: A cluster of 43 questions, produced by graph partitioning. The top three tags covers 90% of the questions in the cluster.

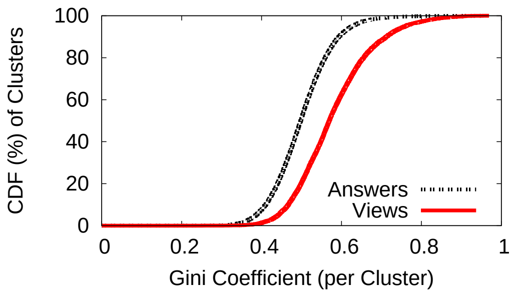


Figure 2.19: Gini coefficient of view (answer) distribution in each cluster.

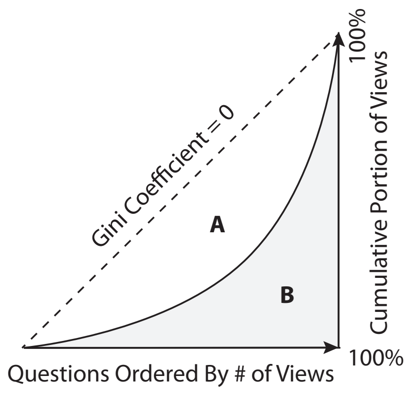


Figure 2.20: Gini coefficient, $G = A/(A + B)$.

table. We see that the three topics are different but all related. In fact, the top three topics cover 90% of the questions in this cluster, which indicates a good cluster focused around a single subject.

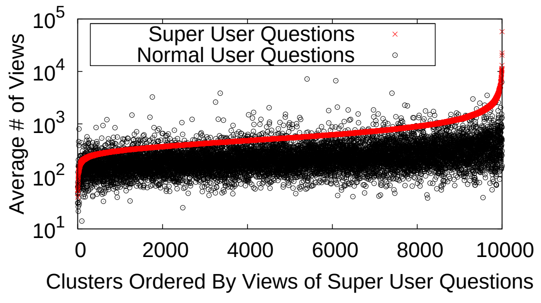
Cluster Analysis. Based on the generated clusters, we can now answer the high level question: do similar questions receive equal attention? We answer this question by assessing the distribution of user views and answers between questions in the same cluster. We choose to use *gini coefficient*, a uniformity metric commonly used to evaluate the equality of distributions in economics [67].

We explain how we compute gini coefficient for each question cluster using Figure 2.20. As an example, the x-axis has the questions sorted by increasing number of views, and the y-axis represents the cumulative portion of the views. So the curve represents y% of contribution (of views) by the bottom x% of questions. By definition, the curve is always at or below the dashed line which represents perfect equality of the distribution. Gini coefficient is defined to quantify how close the curve is to the dashed line: $G = \frac{A}{A+B}$, where A and B represent the corresponding areas above and below the curve. As each axis is normalized to 100%, the gini coefficient G is always within the range of $[0, 1]$, where G=0 means perfect equality or uniformity (the dashed line in our example) and G=1 means an extremely skewed distribution.

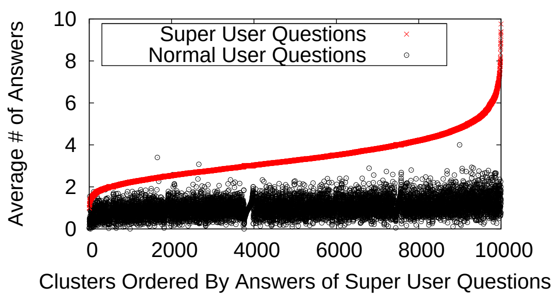
We compute the gini coefficient for the distribution of number of views (and answers) of questions in each cluster. As shown in Figure 2.19, the solid curve shows the gini coefficient of number of views is highly skewed towards 1. More than 90% of clusters have gini coefficient $>$0.4. This shows that the numbers of views are extremely uneven among similar questions within each cluster. The same trend applies to answers, as the vast majority of clusters have extremely skewed answer distributions. This means that user attention is tightly focused on a small portion of (valuable) questions within each cluster of similar questions.

Our results suggest that the structure of the related question graph (*e.g.* question degree) is at least partially responsible for focusing user attention and answers on a small subset in each cluster of related questions. Next, we ask whether super users play a role in directing traffic towards specific questions in each cluster of related questions.

Super User Effect. We evaluate whether the skew in the distribution is caused by super user effect. Intuitively, when a user adds new answers or upvotes existing answers on a question, that question will be pushed to all her followers. Thus super users with more followers can disseminate the question to a larger audience. We use the same definition of super users as in previous analysis by taking the top 5% of most followed users. We measure the super user effect by comparing the number of views (answers) of questions involving super users to other questions with no super user involvement. Among all 10000 clusters, only 1 cluster has no super user in any of its questions, and is not considered in the analysis.



(a) Views



(b) Answers

Figure 2.21: Average # of views (answers) of super user questions vs. normal user questions in each cluster.

Figure 2.21 shows the scatter plot of average views (and answers) of super user involved questions and normal user questions in each cluster. The X-axes are presented in ascending order of the views (answers) of super user questions, thus the super user question points form a near-continuous line. We first compare the average number of user views in Figure 2.21a. In the vast majority of the clusters, the super user questions have more views than that of questions with no super user involvement. There is only a small number of clusters (4%) where normal user questions receive more user views then super user questions.

Figure 2.21b compares the two type of questions with respect to average number of answers per question. The result shows that super user involved questions have significantly more answers than normal user questions. Compared to user views, it shows a even stronger impact of super users on drawing answers. In different clusters, super user questions have an average number of answers ranging from 2 to 10, while questions without super user involvement almost always stays below 2 answers across clusters. Both the number of user views and answers can reflect how much attention each question receives. The result shows choices made by a small number of super users on questions usually affect the focus of attention for the whole community.

In summary, we build the related question graph, and find that it is a relatively stable structure even as new questions are constantly added to the system. We find that high degree questions generally receive more answers and views compared to others. More specifically, the spread of user views and answers within clusters of related questions is extremely skewed towards a small subset of questions. This bias is likely created by the structure of the question graph, and enhanced by super users, as the questions they interact with receive additional views and answers from their followers.

2.1.7 Summary of Results

Community question and answer sites provide a unique and invaluable service to its users. Yet as these services grow, they face a common challenge of keeping their content relevant, and making it easy for users to “find the signal in the noise,” *i.e.* find questions and content that are interesting and valuable, while avoiding an increasing volume of less relevant content.

In this section, we use a data-driven study to analyze the impact of Quora’s internal mechanisms that address this challenge. We find that all three of its internal graphs, a user-topic follow graph, a user-to-user social graph, and a related question graph, serve complementary roles in improving effective content discovery on Quora. While it is difficult to prove causal relationships, our data analysis shows strong correlative relationships between Quora’s internal structures and user behavior. Our data suggests that the user-topic follow graph generates user interest in browsing and answering general questions, while the related question graph helps concentrate user attention on the most relevant topics. Finally, the user-to-user social network attracts views, and leverages social ties to encourage votes and additional high quality answers. As Quora and its repository of data continues to grow in size and mature, our results suggest that these unique features will help Quora users continue find valuable and relevant content.

2.2 Anonymity and Abuse

2.2.1 Introduction

Over the last decade, online social networks (OSNs) such as Facebook, LinkedIn, and Twitter have revolutionized the way we communicate. By formalizing our offline social relationships into digital form, these networks have greatly expanded our capacity for social interactions, both in volume and frequency.

Yet the industry landscape is changing. Content posted on Facebook is now commonly used to vet job candidates, support divorce litigation, and terminate employees. In addition, studies have observed a significant growth in privacy-seeking behavior, even despite changes in social networks to encourage broader information sharing [208]. Finally, these trends have only been accelerated by recent revelations following the Snowden disclosures, with numerous headlines reminding Internet users that their online behavior is under constant scrutiny by NSA and other entities.

All these have contributed to the rapid rise of a new wave of privacy-preserving communication and social networking tools. These fast-growing services are pseudo-anonymous messaging mobile applications: *SnapChat* made headlines for ensuring that photos self-destruct in a few seconds; *Whisper* allows users to anonymously post their thoughts to a public audience; and *Secret* allows users to share content with friends without revealing their own identity. This is just the tip of the iceberg, as many similar services are popping up with increasing frequency, *e.g.*, Tinder, Yik-yak, and Wickr.

The anonymous nature of these communication tools has drawn both strong supporters as well as vocal critics. Supporters believe that they provide valuable outlets for whistleblowers avoiding prosecution, and allow users to express themselves without fear of bullying or abuse [238, 237]. Critics argue that the lack of accountability in these networks enables and encourages negative discourse, including personal attacks, threats, and rumor spreading [42, 40].

Yet all parties agree that these tools have had a dramatic impact on how users interact and communicate.

In this part of the chapter, we describe our experience and findings in our effort to study pseudo-anonymous social networks, through a detailed measurement and analysis of *Whisper*. Whisper is a mobile app that allows users to post and reply to public messages on top of an image (*e.g.* Internet memes), all using anonymous user identifiers. Whisper does not associate any personal identifiable information with user IDs, does not archive any user history, and does not support persistent social links between users. These design choices are the polar opposite of those in networks such as Facebook. Yet they have made Whisper one of the most popular new social networks, with more than 3 billion page views per month[2]. As our working dataset, we captured 100% of the Whisper data stream for a 3-month period starting in February 2014, including more than 24 million whispers and replies written by more than 1 million unique users.

We focus our study on the net impact of anonymity in Whisper, compared to traditional social media with verified identities and social links. Given the large differences between Whisper and current leaders such as Facebook and LinkedIn, our analysis can have significant implications on future infrastructures for social networks, issues of user privacy in messaging networks, and our understanding of social behavior. More concretely, our study also sheds light on the long-term sustainability of anonymous communication networks, given the removal of persistent social links, often considered key to the "stickiness" of today's networks.

Our analysis provides several key findings.

- First, we seek to understand user interactions in the absence of social links. We build interaction graphs and compare them with those of traditional social networks like Twitter and Facebook. Not surprisingly, we find that user communication patterns show high dispersion, low clustering, significantly different from prior systems. Per user, we observe

[2]To our knowledge, there is no public data on Whisper user counts.

that "friends" are highly ephemeral, and strong, long-term friendships are rare.

- Second, our study of user activity over time shows that a constant stream of new users contribute significantly to content generation, and users bifurcate clearly into short-lived (1-2 days) and long-term users. We demonstrate that users can be accurately classified into either group by applying ML techniques to only 1 week's worth of activity history.
- Third, we study the question of abusive content through analysis of "deleted whispers." We show that most deleted whispers focus on adult content, and Whisper's moderation team usually deletes offensive whispers within a short time after initial posting.
- Finally, we identified a significant attack that exposes current Whisper users to detailed location tracking. We describe the attack in detail and our experiments. Note that we have already notified Whisper of this vulnerability, and they are taking active steps to mitigate the problem.

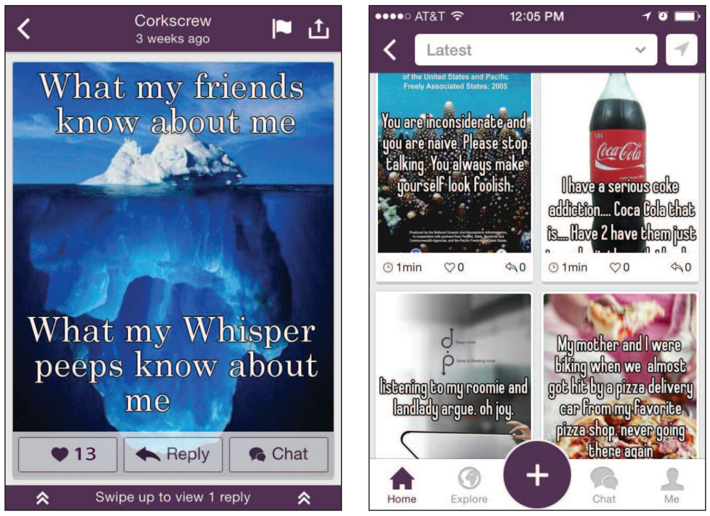
2.2.2 Background: Whisper Network

In this section, we briefly describe background information about the Whisper network, followed by a high level summary of the goals of our study.

2.2.2.1 The Whisper Network

Whisper.sh is a two-year old smartphone app that has become a leader in a new wave of pseudo-anonymous messaging and social communication services, including *Snapchat*, *Secret*, *Tinder*, *Yik-yak*, *Ether* and *Wickr*. While detailed functionality may vary, these services generally provide ways for users to make statements, share secrets or gossip, all while remaining anonymous and untrackable.

As a mobile-only service, Whisper allows users to send messages, receive replies using anonymous nicknames. It has grown tremendously in popularity since launching in 2012, and



What my friends know about me

What my Whisper peeps know about me

You are inconsiderate and you are naive. Please stop talking. You always make yourself look foolish.

I have a serious coke addiction.... Coca Cola that is.... Have 2 have them just

listening to my roomie and landlady argue. oh joy.

My mother and I were biking when we almost got hit by a pizza delivery car from my favorite pizza shop. never going there again

Figure 2.22: Screenshot of a sample whisper message (left) and the public stream of latest whispers (right).

averages more than 3 billion monthly page views as of early 2014 [89]. The functionality is very simple: the app overlays each user's short text message on top of a background image based on keywords from the message (Figure 2.22). The resulting *whisper* is posted to the public with the user's random or self-chosen nickname. Others can *heart* (Whisper's version of "like") a message anonymously, or post a public followup reply with their own whisper. In addition, users can send *private messages* to the author of a whisper to start a chat, and private messages are only visible to the participants.

User Anonymity. Whisper's focus on anonymity breaks some of the core assumptions made in traditional social networks like Facebook or Google+. First, Whisper users are identified only by randomly assigned (or user-chosen) nicknames, not associated with any personal information, *e.g.*, phone numbers or email addresses[3]. Second, Whisper servers only store public Whispers, and users' private messages are only stored on their end user devices. There is no functionality to search or browse a specific user's historical whispers or replies. Third,

[3]On the server side, Whisper associates new users with a globally unique identifier (GUID), and binds it to the DeviceID of user's phone. Users can transfer their accounts (private message history) when switching to new phones via iCloud.

there is no notion of a persistent social link between users (*e.g.*, friends on Facebook, followers on Twitter). Thus users are encouraged to interact with a wide range of strangers instead of a known group of "friends."

Public Feeds. Without social links, users browse content from several public lists instead of the news feed of their friends (or followees). These lists include a *latest* list which contains the most recent whispers (system-wise); a *nearby* list which shows whispers posted in nearby areas (about 40 miles of radius range); a *popular* list which only shows top whispers that receive many likes and replies; and *featured* list which shows a subset of popular whispers that are hand-picked by Whisper's content managers. All these lists sort content by most recent first.

2.2.3 Data and Initial Analysis

Before diving into our analysis of Whisper, we first describe our data collection methodology and collected datasets. We then describe some high level analyses of our dataset.

2.2.3.1 Data Collection

Our goal is to collect whispers and their replies posted in the entire network. Given that Whisper does not archive historical data, our method is to keep crawling newly posted whispers over a long period (February to May 2014). We focus on the "latest" list, which is a public stream of the latest whispers from all Whisper users. Unlike other public lists *e.g.*, "nearby" and "popular", the "latest" list provides access to the entire stream of whispers in the network. Since Whisper does not provide a third-party API, we crawl the "latest" list by scrapping Whisper's website.

Each downloaded whisper includes a whisperID, timestamp, plain text of the whisper, author's GUID, author's nickname, a location tag, and number of received likes and replies. An author's GUID was not intended to act as a persistent ID for each user, but was implemented that way due to Whisper's dependency on a third-party service for private messages. Authors' GUIDs make it possible to track a user's posts over time. After we reported this issue to Whisper's management team, they removed the GUID field in June 2014. The location tag shows user location at the city and state level (*e.g.*, Los Angeles, California), and is available only if the whisper author enabled location sharing permission. Replies to a whisper are similar, the only difference is that replies are also marked with the whisperID of the previous whisper in the thread.

Crawling. We implemented a distributed web crawler with two components, a main crawler that pulls the latest whisper list, and a reply crawler that checks past whispers and collects all sequences of replies associated with an existing whisper. We observe that Whisper servers keep a queue of the latest 10K whispers. Running the main crawler every 30 minutes ensures that we capture all new whispers. In contrast, crawling for replies is more computationally intensive. We crawl for replies every 7 days, and check for new replies for all whispers written in the last month. In practice, we observe that whispers usually receive no followup replies 1 week after being posted.

We ran our crawler from February 6 to May 1, 2014. During this period of roughly 3 months, we collected 9,343,590 total Whispers with 15,268,964 replies and 1,038,364 unique GUIDs. Thanks to server side queues, we collected a continuous data stream despite a small number of interruptions to update crawler code. The only point of note is that, at Whisper's request on April 20, we shifted our crawlers to crawl a different Whisper server using a new set of API calls. The shift reduced load for Whisper, but produced whispers without location tags. Since this only affected 10 days of data, we believe this has little impact on our analysis of location-based features.

Validating Consistency. We further verify the completeness of the "latest" stream using a

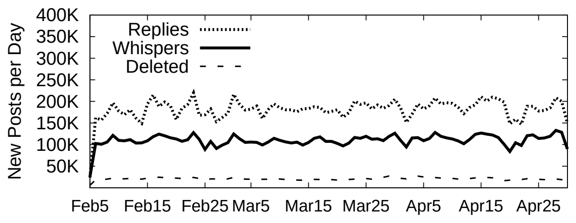


Figure 2.23: Number of new whispers, new replies and deleted whispers each day.

small experiment. We use HTTP requests to simultaneously crawl the “nearby” streams of 6 locations near different cities: Seattle, Houston, Los Angeles, New York, San Francisco and Chicago. We capture these streams for 6 hours, and confirm that the 2000+ whispers from 6 locations were all present in the “latest” stream during the same timeframe.

Limitations. There are two types of data our measurements do not capture. First, we do not capture users who only read/consume whispers but never post any content. Since these passive users do not generate visible user interactions, they are unlikely to affect the majority of our conclusions. Second, our data is limited to visible public data, and we do not have access to private messages between users. Thus our results represent a lower bound on user interactions in the system. As we discuss later, we believe there should be strong correlation between public interactions and private messages.

2.2.3.2 Preliminary Analysis

Next we present some high level results on our dataset of whispers, replies and users. Our results in this section set the context for more detailed analysis on user behavior and anonymity in later sections.

Whispers Over Time. We begin by looking at whisper posts over time. Figure 2.23 shows number of new whispers and replies posted every day during our study. As shown, new content in Whisper is relatively stable, averaging 100K new whispers and 200K replies per day. One interesting observation is that in any time frame, there are significantly more replies than there

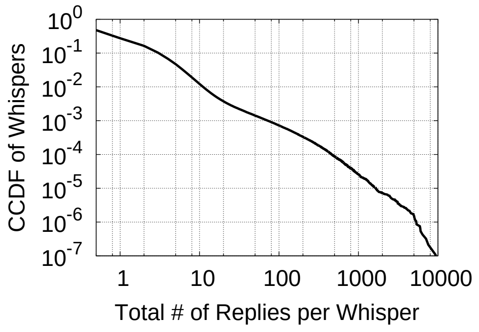


Figure 2.24: Total number of replies per whisper.

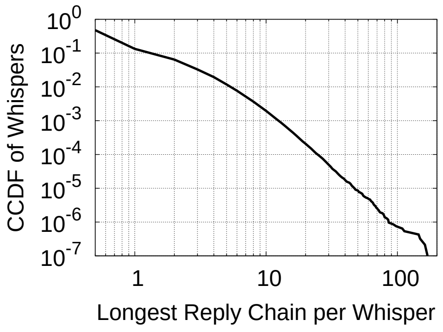


Figure 2.25: Length of longest reply chain per whisper.

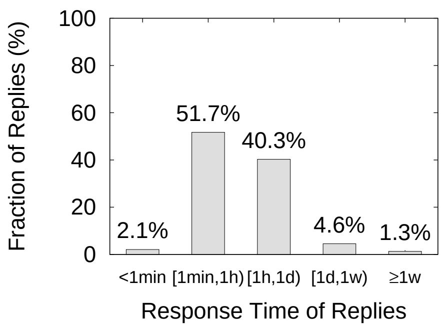


Figure 2.26: Time gap between reply and original whisper.

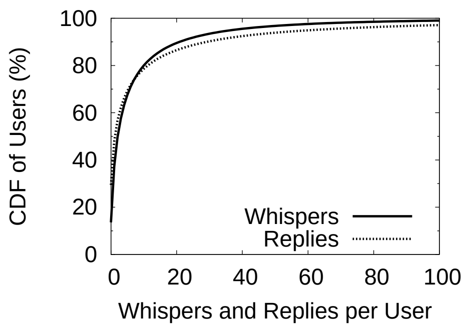


Figure 2.27: Whispers and replies posted per user.

are original whispers.

During our data collection, we found that a significant portion of whispers is deleted by either the author or Whisper moderators. As far as we can determine, old Whispers do not “expire” and stay on Whisper servers, and can be referenced by following a chain of replies. For deleted whispers, however, we receive an “the whisper does not exist” error when we try to re-crawl their replies. Among the 100K new whispers posted every day, roughly 18% are eventually deleted. We analyze deleted whispers in detail later in §2.2.6.

Replies. Users can post replies to a new whisper or other replies. Multiple replies can generate their own replies, thereby forming a tree structure with the original whisper as the root. Figure 2.24 and Figure 2.25 show total replies per whisper and the longest chain length (maximum tree depth) per whisper. Unsurprisingly, 55% of whispers receive no replies. Since all whispers are posted to the same public lists, each whisper only has a short time window to attract users' attention. Among whispers with replies, roughly 25% have a chain of at least 2 replies. These essentially become threads of conversations between users.

Figure 2.26 plots the distribution of reply arrival time, which is the time gap between each reply and the original whisper. 54% of replies arrive within an hour of the original whisper, and more than 94% of replies arrive within a day. Only 1.3% of replies arrive a week or more after the whisper. This confirms our intuition—if a whisper does not get attention shortly after posting, it is unlikely to get attention later.

Users. We look at content-generated per user based on unique GUID. Figure 2.27 plots the number of whispers and replies posted by each user. Most users (80%) post less than 10 total whispers or replies. Roughly 15% of users only post replies but no original whispers, and 30% of users only post whispers but no replies.

Content Analysis. A high-level analysis of the contents of whispers shows that users post highly personal content. A search of singular first-person pronouns (*e.g.*, I, me, my, myself) hits about 62% of all whispers. We also find a heavy usage of emotional key words. Specifically, 40% of whispers contain one of the 1,113 human mood related key words provided by WordNet Affect [205]. Finally, people often ask questions seeking advice or empathy. About 20% of whispers are questions, based on the usage of question marks and interrogatives (*e.g.*, what, why, which). These three categories effectively cover 85% of all whispers. It is clear that the anonymity provided by Whisper encourages users to post personal and intimate content without privacy concerns. We will take a closer look at "topics" of whispers in §2.2.6.

2.2.4 User Interactions

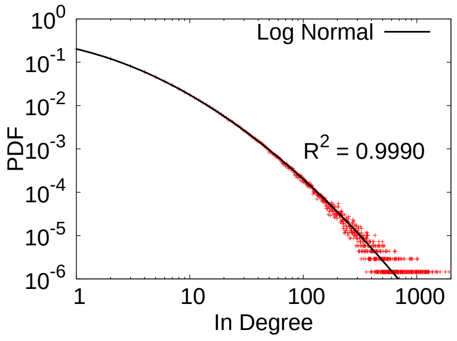
Our study begins with user interactions on Whisper. The fact that Whisper users cannot construct persistent social links between them fundamentally changes how users interact and develop friendships. In this section, we study the bidirectional interactions built from whispers and their replies, and seek to understand user interactions from three different levels. First, we study interactions at a global network level, by comparing structural properties of the Whisper interaction graph to those of traditional OSNs, *e.g.*, Facebook and Twitter. Then, we look at interactions at per-user level to understand if users still develop strong ties (frequently interacted friends) in Whisper.

2.2.4.1 Whisper Interaction Graph

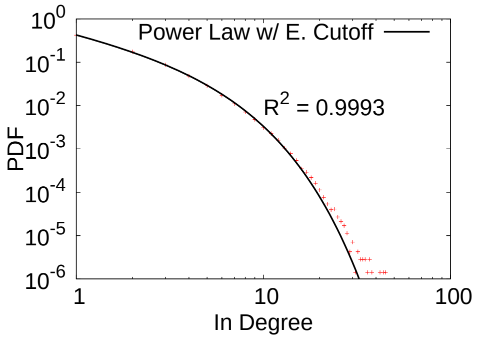
We first compare the interaction graph of Whisper with those of traditional online social networks (Facebook and Twitter). Our goal is to understand whether the lack of social links in Whisper fundamentally changes users' interaction patterns at an aggregate network level. We build a Whisper interaction graph based on whispers and replies, and compare its structure to those of graphs constructed from Facebook wall posts and Twitter retweets.

Building Interaction Graphs. We build the Whisper interaction graph based on whispers and followup replies, which are the primary publicly visible interactions in Whisper. The result is a directed interaction graph, where nodes are users and edges represent reply actions. For example, if user A posts a reply whisper to B's whisper, we build a directed edge from A to B. Only direct replies are used to build edges. We remove disconnected singleton nodes from the graph. We produce a main interaction graph from our 3 month dataset (*Whisper-all*).

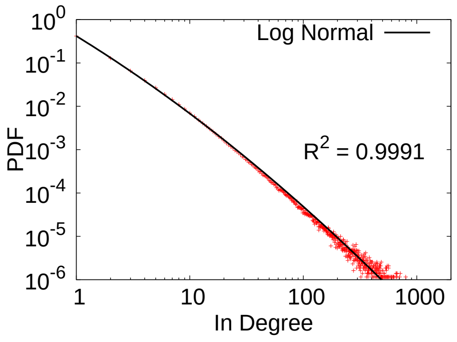
For comparison, we also build interaction graphs for Facebook and Twitter, based on anonymous datasets from our prior work [236, 239]. Both datasets crawled historical data that covers user interactions over at least 3 months. We built a directed interaction graph using



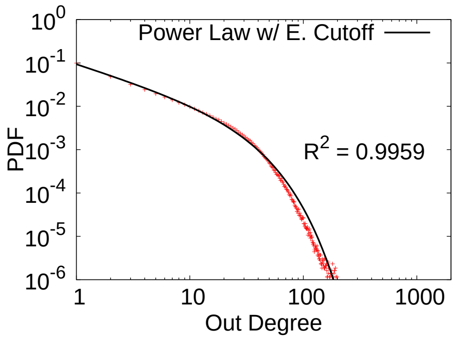
(a) Whisper In



(b) Facebook In



(c) Twitter In



(d) Twitter Out

Figure 2.28: Degree distribution and fitting result.

Graph	# Nodes	# Edges	Degree	C. Coef.	Path Len.	Assort.	SCC/WCC
Whisper	690K	6,531K	9.47	0.033	4.28	-0.011	63.3% / 98.9%
Facebook	707K	1,260K	1.78	0.059	10.13	0.116	21.2% / 84.8%
Twitter	4,317K	16,972K	3.93	0.048	5.52	-0.025	14.2% / 97.2%

Table 2.4: High level statistics of different interaction graphs.

Facebook wall post data: if user A posts on user B's wall, we create a directed edge from A to B. For Twitter, we built the graph based on retweet interactions: if user A retweets a tweet from B, we create a directed edge from A to B. To match the 3-month time coverage of Whisper graph, we build similar Facebook and Twitter graphs each using data covering 3 month periods. Table 2.4 shows the key statistics of all three interaction graphs.

Degree Distribution and Fittings. Users in Whisper show much higher average degree than users in Facebook and Twitter, meaning users interact with a larger sample of other users. We determine the best fitting function for each graph's degree distribution using 3 commonly used fitting functions for social graphs, power law ($P(k) \propto k^{-\alpha}$), power law with exponential cutoff ($P(k) \propto k^{-\alpha}e^{-\lambda k}$) and lognormal ($P(k) \propto e^{\frac{(lnx-\mu)^2}{2\sigma^2}}$) [86, 236]. We follow the fitting method in [65] and use Matlab to compute fitting parameters and accuracy (R-squared values), and show the results in Figure 2.28. For both the Whisper and Facebook graphs, the out-degree distribution looks similar to the in-degree distribution. For brevity, we only show the in-degree distribution for each graph. Intuitively, Facebook was designed to emulate offline social relationships, and the prevalent bidirectional interactions lead to symmetric in- and out-degree distributions. For Whisper, user interactions are largely random between users. In contrast, Twitter's in-degree and out-degree distributions are significantly different. It's well known that Twitter is more of an information dissemination medium than a social network, and interactions are highly asymmetric [128].

Clustering Coefficient. Clustering coefficient is the ratio of the number of connections that exist between a node's immediate neighbors over all possible connections that could exist. It measures the level of local connectivity between nodes. Clustering coefficient in the Whisper graph (0.033) is much smaller than that of Facebook (0.059) and Twitter (0.048). The cause is clear: Whisper users are highly likely to interact with complete strangers, who are highly unlikely to interact with each other.

Average Path Length. Average path length is the average of all pairs of shortest paths in the graph. Given the size of our graphs, it's impractical to compute the shortest path for all node pairs. Instead, we randomly select 1000 nodes in each graph and compute the average shortest path from them to all other nodes in the graph. The result shows that Whisper graph has the shortest average path length of the 3 networks. This is again intuitive, since the formation of interactions between random strangers creates numerous shortcuts in the graph, thereby shrinking the average path length. Considering Whisper's high average degree, low clustering level and short average path length, Whisper exhibits more properties of a random graph [234]

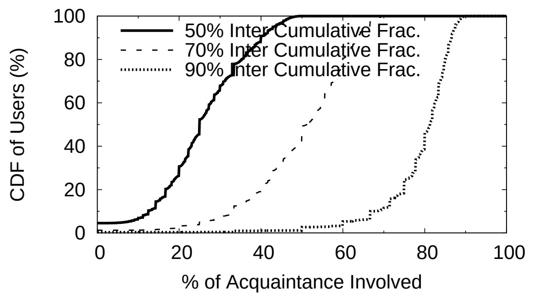


Figure 2.29: The distribution of users' interaction among their acquaintances, for different % of interactions.

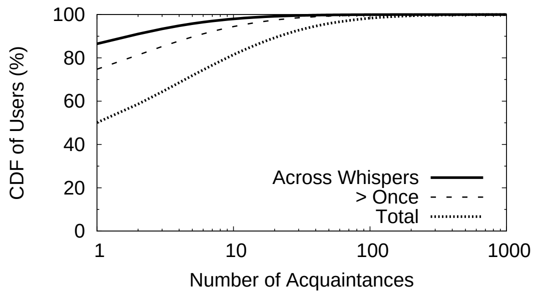


Figure 2.30: Number of user's acquaintances, and those that users interact $>$ once and across whispers.

than those of a "small-world" network like Facebook and Twitter.

Assortativity. Assortativity coefficient measures the probability for nodes in a graph to link to other nodes of similar degrees. Assortativity > 0 indicates that nodes tend to connect with other nodes of similar degree, while assortativity < 0 indicates that nodes connect to others with dissimilar degrees. Our result shows the assortativity coefficient of Whisper graph is very close to zero (-0.011), which closely resembles a random graph [163]. In contrast, similar users tend to flock together in social networks with bidirectional links (*e.g.*, Facebook), producing positive assortativity (0.116). In Twitter, large numbers of normal users follow celebrities and notable figures, thus producing a more negative assortativity (-0.025).

2.2.4.2 User Interactions and Strong Ties

Finally, we analyze user interactions and implicit social links at the per-user level. Recall that Whisper's lack of persistent identities and social links encourages users to interact with strangers. In the following, we seek the answers to two key questions. First, do users have a fixed set of "friends" that they frequently interact with? Such friendships could have formed despite the anonymous nature of Whisper nicknames. Second, how likely are any strong ties the result of offline friendships?

Per-user Interaction. We search for potential friendships (*i.e.* strong ties) by looking for pairs of users who interact more frequently with each other than with others. For convenience, we call the set of people that a user interacts with (regardless of direction) as her *acquaintances*. For each user, we compute a distribution of her interactions across her top acquaintances, and look for *skew* in her interactions with all acquaintances.

We select several points (50-, 70- and 90-percentiles) from each user's distribution and aggregate them in a CDF to show the percentage of top acquaintances involved (Figure 2.29). To avoid statistical outliers, we only include users with at least 10 interactions.

We find user's interactions are distributed rather evenly across acquaintances. Take the 90-percentile line for example, for nearly all the users (∼90%), more than 70% of their acquaintances are responsible for 90% of their interactions. This relatively low skew in Whisper is exactly the opposite of traditional OSNs like Facebook, where a small fraction of friends (strong ties) are responsible for the vast majority of user's interactions [236].

Interaction across Whispers. Across a user's acquaintances, we look for potential strong ties, *i.e.* acquaintances with whom the user interacts often. Figure 2.30 shows user's number of total acquaintances, acquaintances that users interacted more than once, and acquaintances that users interact more than once *using multiple whisper threads*. In Whisper, it's common for people to interact more than once under the same whisper. However, it's rare to talk with the same person across different whispers, because keeping track of a particular user via their anonymous nickname is difficult. As shown in Figure 2.30, only 13% of users have acquaintances that they interact with across whispers.

We then select those user pairs who have interacted across whispers for further analysis. In total, there are 503K such user-pairs. Figure 2.31 presents the heat map of these user-pairs' lifespan (timespan between their first and last interaction) and their number of interactions across whispers. Note that the color palette is log-scale—the vast majority of user pairs are stacked at the left bottom corner, indicating short-lived, low-interaction relationships. Only a

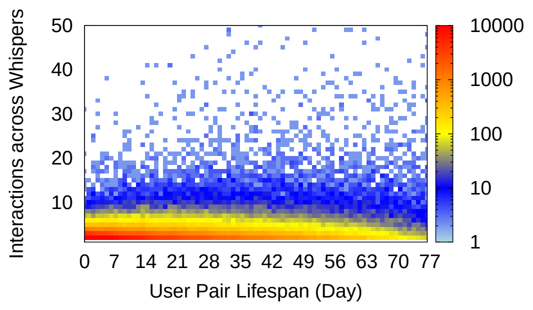


Figure 2.31: Users pairs with interaction across whispers: lifespan vs. # of interactions.

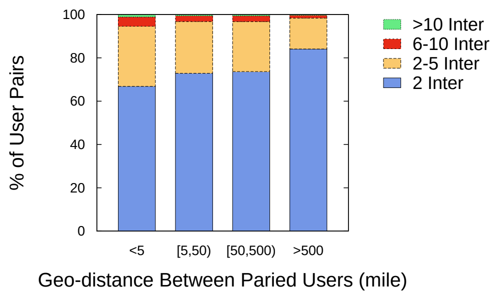


Figure 2.32: For all user pairs, distance between two users vs. # of interactions of the user pair.

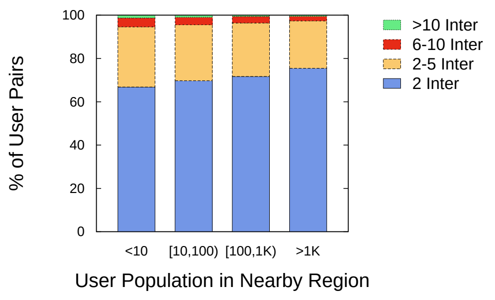


Figure 2.33: For nearby user pairs, user population in nearby areas vs. # of interaction of the user pair.

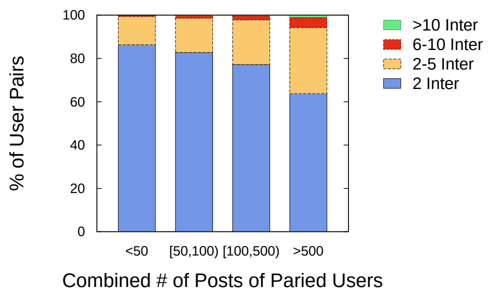


Figure 2.34: For nearby user pairs, total # of whispers vs. # of interactions of the user pair.

very small fraction of outliers (right top corner) achieved long-term and frequent interactions.

Friends or Random Encounters? Even though the strong ties are outliers, it is interesting to explore how could these user-pairs constantly interact with each other *across whispers*: Are these pairs of offline friends who actively track each other in the public feeds (using nicknames), or are these simply users who bump into each other often by chance? We realize this is a very hard question to answer deterministically. But we have a key intuition: if these interactions are truly random, then it is highly likely that these two users are co-located in same geographic area, particularly areas with a sparse population of Whisper users. Then as long as the two users actively post whispers, they have a good chance to see each other in the nearby list.

Now we use our data to test this intuition. For user-pairs with cross-whisper interactions, we first examine their geographic distances[4]. We find that among 503K user pairs, 90% have two users co-located in the same "state" and 75% have their distance <40 miles which is the maximum range of the nearby stream. Figure 2.32 shows the correlation between geo-distance and the interaction frequency of user pairs. Each stacked bar adds up to 100%, and each category represents user pairs with different interaction level (*i.e.* number of interactions across whispers). It shows that frequent interactions are more skewed to users that are geographically close to each other.

Then we further examine these pairs co-located in nearby areas (*i.e.* distance <40 miles). More specifically, we analyze two factors that potentially impact users' likelihood of chance encounters—the user population in the geographic area and total number of whispers posted by the two users (Figure 2.33 and Figure 2.34). Intuitively, the smaller user population in the same nearby area, the higher chance to encounter the same person in the nearby list again and again. Similarly, the more whispers two users post, the more likely they encounter each other and form interactions. Here the user population is estimated by the total number of unique users that have the same city-level location tags with the paired users. Both results confirm our intuition. As user population density decreases and as the number of user posts increases, the probability of more frequent user-pair interactions also increases.

In summary, our analysis suggests strong ties are extremely rare in Whisper. We also find strong ties are skewed to user-pairs who have a higher chance to encounter each other (*i.e.* active users that are co-located in areas with sparse user population). Thus while it is possible to develop strong relationships from Whisper interactions, such relationships are likely heavily influenced by geographic density and user whisper frequency. Note that our analysis relies on only public interactions and do not include private messages. Intuitively, we believe

[4]This is the distance between two user's city-level tags. The GPS coordinates of each city are obtained from Google Geocoding API.

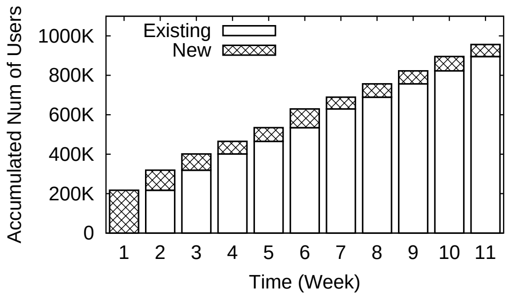


Figure 2.35: The growth of user population in our dataset over time.

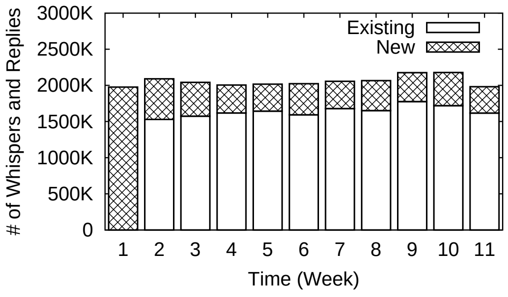


Figure 2.36: # of whispers and replies by new and old users per week.

users' private interactions should correlate with their public interactions, and we can predict user pairs with private interactions from their public interactions. Prior work also confirms that public interactions are more informative when modeling strength of ties than private communications [117, 83].

2.2.5 User Engagement

Thus far our analysis shows Whisper users tend to interact with strangers rather than stable friends. The negative consequence is that a lack of strong ties usually produces a less "sticky" network, *i.e.* fewer disincentives to prevent users from leaving [78]. This raises a natural question: without strong ties, can Whisper users stay engaged in the network in the long run?

In this section, we seek to consider this question by looking at *per-user* engagement. First, we examine user engagement over time to understand user attrition in the 3 month period of our dataset. Second, we evaluate a machine learning classifier and show that we can accurately predict whether users stay engaged in the system using only a short history of their actions after their first post. We use experiments to determine key signals that indicate a user's intention to leave. Note that our analysis is limited to "active" users who have posted at least 1 whisper or reply, and does not include passive users who consume but do not contribute content.

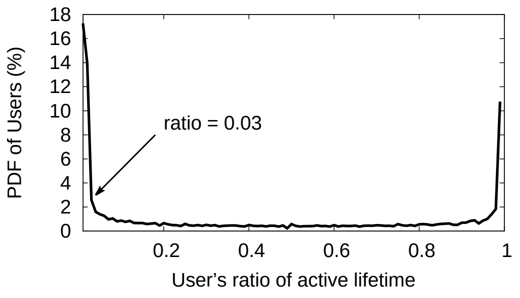


Figure 2.37: User's active lifetime over staying time in our dataset.

2.2.5.1 User Engagement Over Time

We start with basic analysis of user activity over time using three metrics: user population growth, content contribution by new versus existing users, and the distribution of users' active lifetime.

User Population Growth. Figure 2.35 shows the total number of users over time (11 weeks) in our dataset. Each bar shows a breakdown of new users who just joined that week (new) and the existing users we observed before that week (existing). We observe a stable arrival rate of new users to the network, roughly 80K new users per week. Recall that the daily new posts (whispers and replies) in the entire network remain roughly stable (see Figure 2.23), despite the growth in users. This indicates there are an ongoing number of users who "disengage," *i.e.* stop posting whispers or replies.

Content by New and Existing Users. This motivates us to look at the relative contribution of content by new and existing users. Figure 2.36 shows the breakdown of posts (whispers plus replies) by users who showed up for the first time in the current week (new) and users who showed up before this week (existing). We find that new users make significant contributions to the overall whisper stream ($> 20\%$). However, as more and more users transition from new to "existing users," content generation by existing users does not grow significantly over time. This confirms our intuition that a certain portion of users are disengaging over time.

Per-user Active Period. Next, we focus on individual users and examine how long users stay active before they disengage. More specifically, we compute their active "lifetime" (timespan between their first and last posts) over their staying time in the dataset (timespan between a user's first post and the last date of our data collection). Given our focus on long-term activity, we exclude users who just recently joined during the last the month of our data collection. Thus for Figure 2.37, we only consider users who have been in our dataset for at least one month (70.3% of all users).

Figure 2.37 shows the distribution of user's ratio of active lifetime (PDF). Users are clearly clustered into two extremes: one major cluster around an extremely low ratio (0.03), representing those who quickly turned inactive in 1 or 2 days after their first post; another major cluster around 1.00, representing users who remain active for their entire time in the dataset (at least 1 month). Similar patterns have also been observed in other user generated content (UGC) networks, such as blogs and Q&A services [92]. If we set a threshold for active ratio at 0.03, these "try and leave" users account for 30% of all users. This explains our observation in Figure 2.36—because a significant portion of users become inactive quickly, the overall content posting rate remains stable despite a significant number of new users joining the network.

2.2.5.2 Predicting User Engagement

A key observation of the above analysis is that Whisper users tend to fall into one of two behavioral extremes—either staying active for a long time, or quickly turning inactive (Figure 2.37). The bimodal nature of the distribution hints at the potential to classify users into the two clusters.

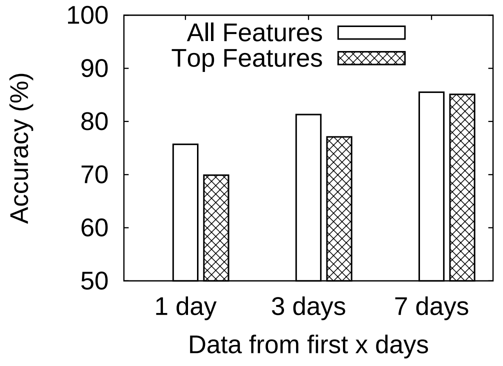
Here, we experiment with machine learning (ML) classifiers to determine if we can predict long term user engagement based on their early behavior after their first post (in our dataset). We seek to answer three key questions: First, is this prediction even possible? Second, what ML models produce the most accurate predictions? Third, what early-day signals can most strongly indicate a user's intention to leave?

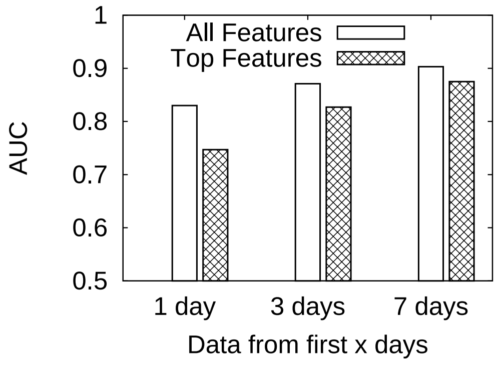
We take three steps to answer the above questions. First, we collect a set of behavioral features based on users' activities in their first X days on Whisper, ideally with a small value for X. Second, we use these features to build different machine learning classifiers to predict long term user engagement. Finally, we run feature selection to determine the features that provide the best early signals indicating which users might disengage.

Features. We explore multiple different classes of features (20 features in all) to profile users' behavior during their first X days. Out of these, we will select the most essential features.

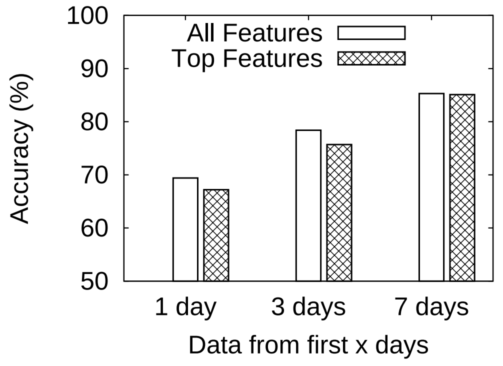
- *Content posting features (F1-F7).* 7 features: user's number of total posts, number of whispers, number of replies, number of deleted whispers, and number of days with at least one post/whisper/reply.
- *Interaction features (F8-F15).* 8 features: ratio of replies in total posts, number of acquaintances, number of bi-directional acquaintances, outgoing replies over all replies, maximum number of interactions with the same user, ratio of whispers with replies, and average number of replies and likes per whisper.
- *Temporal features (F16-F17):* 2 features: average delay before first reply to user's whisper; average delay of user's replies to other users' whispers.
- *Activity trend (F18-F20):* 3 features: we equally split each user's first X days into three buckets and record the number of posts in each bucket ($First$, $Middle$ and $Last$). We compute 2 features as $\frac{Middle}{First}$ and $\frac{Last}{First}$. Finally, whether the number of posts decreases monotonically across the three buckets.

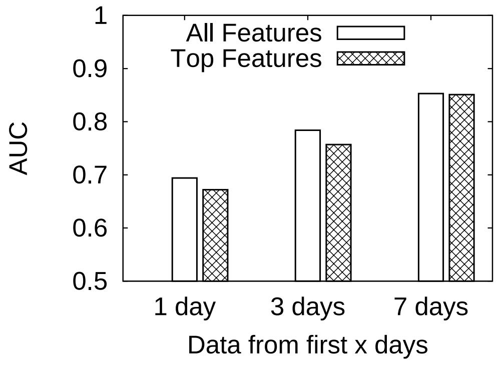
Classifier Experiments. To build a training set for our classifiers, we focus on users that have at least a month's worth of activity history in our dataset (730K users). We select a set of "short-term" users who tried the app for 1-2 days and quickly disengaged (no more





(a) Predicting Inactive vs. Active (RF)





(b) Predicting Inactive vs. Active (SVM)

Figure 2.38: Prediction result using Random Forests and SVM. The model performance is evaluated by accuracy (left) and Area under ROC curve (right) .

posts). Using results from Figure 2.37, we randomly sample 50K users from those whose active lifetime ratio < 0.03 as the *Inactive* set. We then choose a random sample of 50K users whose active lifetime ratio > 0.03 to form the *Active* set.

Our goal is to classify the two sets of users solely based on users' activities in their first X days, and we use 1, 3 and 7 as values of X. We build multiple machine learning classifiers including Random Forests (RF), Support Vector Machine (SVM) and Bayes Network (BN), using implementations of these algorithms in WEKA [94] with default parameters. For each experiment, we run 10-fold cross validation and report classification accuracy and area under ROC curve (AUC). Accuracy refers to the ratio of correctly predicted instances over all instances. AUC is another widely used metric, with higher AUC indicating stronger prediction

Rank	Observation Time Frame		
	1 day	3 days	7 days
1	Interact-F9 (0.15)	Post-F5 (0.27)	Post-F5 (0.46)
2	Interact-F11 (0.12)	Trend-F19 (0.18)	Post-F6 (0.31)
3	Interact-F10 (0.11)	Post-F6 (0.18)	Trend-F19 (0.28)
4	Interact-F12 (0.11)	Interact-F9 (0.16)	Post-F1 (0.27)
5	Trend-F18 (0.05)	Post-F1 (0.16)	Post-F7 (0.23)
6	Interact-F15 (0.04)	Post-F7 (0.13)	Trend-F20 (0.21)
7	Post-F1 (0.04)	Interact-F15 (0.12)	Interact-F15 (0.21)
8	Interact-F8 (0.04)	Interact-F11 (0.12)	Post-F2 (0.19)

Table 2.5: The top 8 feature and its categories ranked by information gain (values shown in parentheses).

power. For instance, AUC $>$ 0.5 means the prediction is better than random guessing.

The experiment results with Random Forests and SVM are shown in Figure 2.38. The Bayesian results closely match those of SVM, thus we omit them for brevity. We make two key observations. First, behavioral features are effective in predicting future engagement. The accuracy is high (75%) even when only using users' first-day data (RF). This confirms that users' early actions can act as indicators of their future activity. If we include a week's worth of data, we can achieve accuracy up to 85%. Second, we find different classifiers achieve similar performance given 7 days of data. However, their results diverge when they are constrained to using less data (*e.g.*, 1-day). With less data, Random Forests produce more accurate predictions than SVM and Bayesian networks.

Feature Selection. Finally, we seek to identify the most powerful signals to predict a user's long-term engagement. To find the answer, we perform feature selection on the 20 features. More specifically, we rank features based on *Information Gain* [93], which measures feature's distinguishing power over the two classes of data. We list the top 8 features in Table 2.5. As expected, prediction power varies significantly, and information gain drops off quickly (particularly for 1 day) after the top 4 features. To validate their prediction power, we repeat each experiment with only their top 4 features. The results in Figure 2.38 show that the top 4 features achieve most of the accuracy of the entire classifier, but with much less complexity.

Then we take a closer look at the top features. First, we note that the 1-day classifier relies on different set of features compared with 3- and 7-day classifiers. The 1-day models rely heavily on *interaction features*. Intuitively, the model predicts whether a user will stay engaged based on how actively the user participates in social interactions. If a user received many replies or actively replied to others on her first day, there's a high chance for this user to stay longer. For 3- and 7-day models, we find that the key features shift to user's *content posting* and *activity trend* features. This means once we monitor the users for a longer period, the user's intention to stay or leave can be more accurately reflected in her posting frequency and volume, and whether that activity is declining over time.

Engaging Users with Notifications. Stimulating user engagement is a key goal for any new service. One tool Whisper has already deployed is push notifications that deliver the "whisper of the day" to users' mobile device every evening between 7 and 9pm. The exact notification time varies each day and between Android and iOS devices. To examine the impact of these notifications, we conduct a small experiment. We monitor the notification time on 5 different phones every day for 6 days. We look at user activity in the Whisper stream for 5 minute and 10 minute intervals following the notifications, and find no statistically significant increase in new replies or whispers compared to other 5 or 10 minute windows between 7 and 9pm. This means that while these notifications may serve to engage users to read popular whispers, there is no significant increase in new whispers or replies as a result.

2.2.6 Content Moderation in Whisper

Anonymity facilitates free speech, but also inevitably fosters abusive content and behavior [210, 103]. Like other anonymous communities, Whisper faces the same challenge of dealing with abusive content (*e.g.*, nudity, pornography or obscenity) in their network. In addition to a crowdsourcing-based user reporting mechanism, Whisper also has dedicated employees to moderate whispers [89]. Our basic measurements (§2.2.3.2) also suggest this has a significant impact on the system, as we observed a large volume of whispers (>1.7 million) has been deleted during the 3 months of our study. The ratio of Whisper's deleted content (18%) is much higher than traditional social networks like Twitter (<4%) [38, 175].

In this section, we take a closer look at content deletions in Whisper. First, we analyze the content of deleted whispers to infer the reasons behind deletions. Second, we analyze the lifetime of deleted whispers to understand how fast do whispers get deleted. Third, we focus on authors of deleted whispers and compare their behavior to the norm.

Before we begin, we note that while users can delete their own whispers, we believe server-side content moderation is responsible for the large majority of missing whispers in our data. Intuitively, users who reconsider and later delete their own whispers are likely to do so within a relatively short time frame. In contrast, our "deleted" dataset comes from our followup crawl for replies, which runs once a week. In fact, since our main crawler on the latest stream runs every 30 minutes, we expect most self-deleted whispers will not even show up in our core dataset.

Content Analysis of Deleted Whispers. To explore the reasons behind deletion, we analyze the content of deleted whispers. Since whispers are usually very short, Natural Language Processing (NLP) tools do not work well (we confirmed via experiments). Thus we take a keyword-based approach: we extract keywords from all whispers and examine which keywords correlate with deleted whispers. First, before processing, we exclude common stopwords[5] from our keyword list. Also to avoid statistical outliers, we exclude low frequency words that appear in less than 0.05% of whispers. Then for each keyword, we compute a *deletion ratio* as the number of deleted whispers with this keyword over all whispers with this keyword. We rank keywords by deletion ratio, and examine the top and bottom keywords.

We run this analysis on all 9 million original (not including replies) whispers in our dataset,

[5]`http://norm.al/2009/04/14/list-of-english-stop-words`

Topic	**Top 50 Keywords Most Related to Deleted Whispers**
Sexting (36)	sext, wood, naughty, kinky, sexting, bj, threesome, dirty, role, fwb, panties, vibrator, bi, inches, lesbians, hookup, hairy, nipples, freaky, boobs, fantasy, fantasies, dare, trade, oral, takers, sugar, strings, experiment, curious, daddy, eaten, tease, entertain, athletic
Selfie (7)	rate, selfie, selfies, send, inbox, sends, pic
Chat (7)	f, dm, pm, chat, ladys, message, m
Topic	**Top 50 Keywords Least Related to Deleted Whispers**
Emotion (17)	panic, emotions, argument, meds, hardest, fear, tears, sober, frozen, argue, failure, unfortunately, understands, anxiety, understood, aware, strength
Religion (10)	beliefs, path, faith, christians, atheist, bible, create, religion, praying, helped
Entertain. (8)	episode, series, season, anime, books, knowledge, restaurant, character
Life story (6)	memories, moments, escape, raised, thank, thanks
Work (5)	interview, ability, genius, research, process
Politics (1)	government
Others (3)	exactly, beginning, example

Table 2.6: Topics of top and bottom 50 keywords related to whisper deletion.

1.7M of which are later deleted. This produces 2324 keywords ranked by deletion ratio. We list the top and bottom 50 keywords in Table 2.6 and classify them manually into topic categories. Not surprisingly, many deleted whispers violate Whisper's stated user policies on sexually explicit messages and nudity. In contrast, topics related to personal expression, religion, and politics are least likely to be deleted.

Deletion Delay. Next we analyze the deletion delay of whispers, *i.e.* how long do whispers stay in the system before they are deleted? Recall that our reply crawler works once a week, and thus detects deleted whispers on the granularity of once a week. As shown in Figure 2.39, the majority (70%) of deleted whispers are "deleted" within one week after posting. A small portion (2%) of whispers have stayed for more than a month before deletion. Since most whispers lose user attention after one week (Figure 2.26), we believe these deletions are not the results of crowdsourcing flagging, but deleted by Whisper moderators.

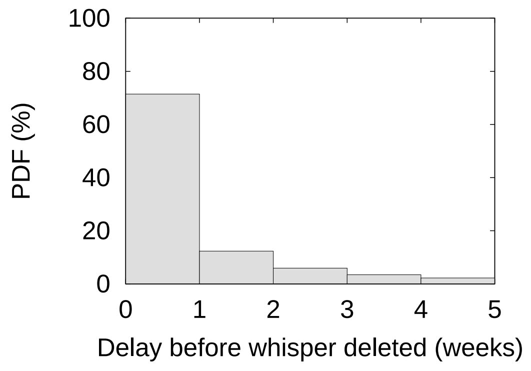


Figure 2.39: Deletion speed (coarse-grained).

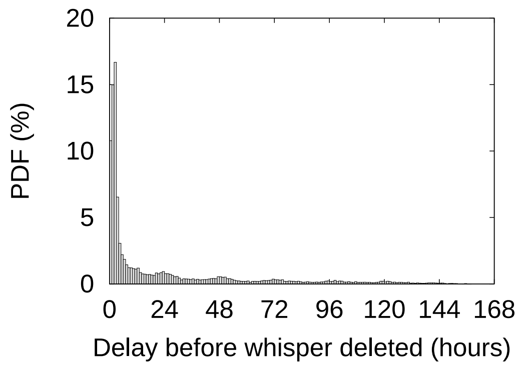
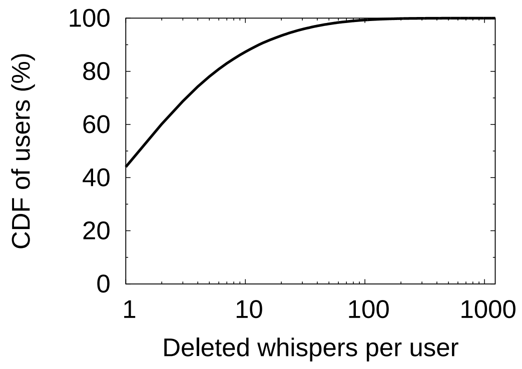


Figure 2.40: Deletion speed (fine-grained).

To get a more fine grain view of whisper deletions, we perform a period of frequent crawls on a small set of whispers. On April 14, 2014, we select 200K new whispers from our crawl of the latest whisper stream, and check on (recrawl) these whispers every 3 hours over a period of 7 days. Of the 200K whispers, 32,153 whispers are deleted during our monitoring period (a week). The more fine-grained distribution of the lifetime (hourly) of these whispers is shown in Figure 2.40. We find the peak of whisper deletion to be between 3 and 9 hours after posting, and the vast majority of deletions happen within 24 hours of posting. This suggests that the moderation system in Whisper works quickly to flag and remove offensive whispers. However, it is unclear whether this level of responsiveness is sufficient, since user page views focus on the most recent whispers, and moderation after 3 hours is possibly too late to impact the content most users see.

Characterizing Authors of Deleted Whispers. Finally, we take a closer look at the authors of deleted whispers to check for signs of suspicious behavior. In total, 263K users (25.4%) out of all users in our dataset have at least one deleted whisper. The distribution of deleted whispers is highly skewed across these users: 24% of users are responsible for 80% of all deleted whispers. The worst offender is a user who had 1230 whisper deleted during the time period of our study, while roughly half of the users only have a single deletion (Figure 2.41).



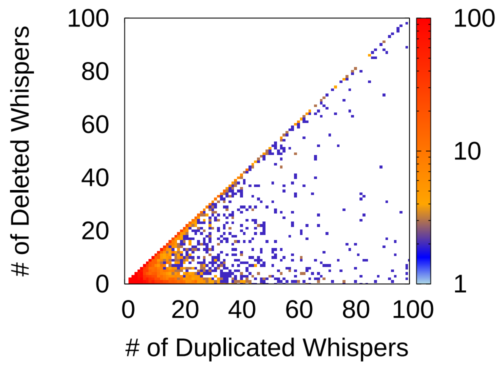


Figure 2.41: # of Deleted whispers per user. Figure 2.42: Duplicated vs. deleted whispers.

We observed anecdotal evidence of duplicate whispers in the set of deleted whispers. We find that frequently reposted duplicate whispers are highly likely to be deleted. Among our 263K users with at least 1 deleted whisper, we find 25K users have posted duplicate whispers. In Figure 2.42, we plot each user's number of duplicated whispers versus the number of deleted whispers. We observe a clear clustering of users around the straight line of $y = x$. This indicates that when users post many duplicated whispers, there's a higher chance that most or all duplicated whispers are deleted.

We also observe that authors of deleted whispers change their nicknames more often than the average user. Figure 2.43 shows the distribution of total number of nicknames used by each user. We categorize users based on how many deletions they have, and also include a baseline of users with 0 deletions. We find users with no deletion rarely change their nicknames, if ever, but nickname changes occur far more frequently for users with many deleted whispers. We speculate that perhaps users change their nickname to avoid being flagged or blacklisted. Since users cannot see their own GUID when using the app, they may assume the system identifies them using only their nickname.

2.2.7 Tracking Whisper Users

In the final component of our Whisper study, we take a close look at a vulnerability that exposes detailed location of Whisper authors to the system. In practical terms, this attack allows a Whisper user to accurately track (or potential stalk) another Whisper user through whispers they've written, by writing simple scripts that query Whisper servers. This attack demonstrates the inherent risks to user privacy in mobile applications, even for apps that target user anonymity as a core goal. Note that we met the Whisper team in person and informed them of this attack. They are supportive of this work, and have already taken steps to remove this vulnerability.

In this section, we describe details of this location tracking attack. The attack makes use of Whisper's "nearby" function, which returns a list of whispers posted nearby, attaching a "distance" field to each whisper. The attack generates numerous "nearby" queries from different vantage points, and uses statistical analysis to reverse engineer the whisper author's location. We validate the efficacy of this attack through real-world experiments.

2.2.7.1 Pinpointing User Locations

We start by describing the high-levels of the attack: when a user (*i.e.* the victim) posts a new whisper, he exposes his location to the Whisper server. An attacker in an nearby area can query the nearby list to get their "distance" to the whisper author. The methodology is simple: the attacker can move to different (nearby) locations and query the nearby list for the distance to the victim. Using multiple distance measurements, the attacker can *triangulate* the whisper author's location. The fact that Whisper does not authenticate location in its queries makes this easier, an attacker can issue numerous distance queries from different locations all while sitting in the comfort of her living room.

With a bit more effort, an attacker can even track the victim's movement over time, by

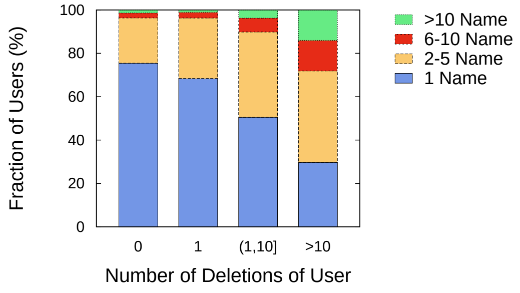


Figure 2.43: User's number of deletions vs. number of nicknames.

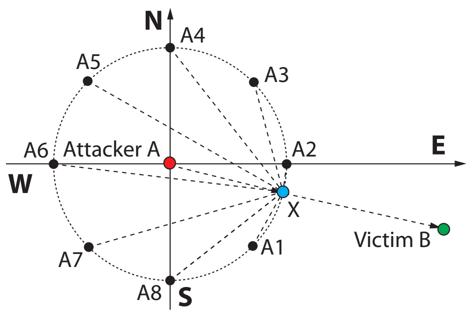


Figure 2.44: Estimating the distance and direction to the victim.

triangulating his location every time he posts a whisper. In practice, this means the attacker can physically go and stalk the victim. While the effective error is roughly 0.2 miles (details below), it is more than sufficient to infer the victim's movement to specific points of interest. Considering most Whisper users are young adults or teenagers [42], this attack can lead to severe consequences.

Distance Granularity and Errors. Implementing this attack is nontrivial. Whisper's design team has always been aware of location tracking risks to its users, and built in basic defense mechanisms into the current system. First, they apply a distance offset to every whisper, so the location stored on their servers is always off by some distance to the actual author location. Second, the distance field returned by the nearby function is a coarse-grained integer value (in miles). This was a recent change made by Whisper in February 2014, before which the nearby function returned distances with decimal values. Third, Whisper server adds a random error to the answer to each query, *i.e.* when we query the nearby list repetitively from the same location, each query returns a different distance for the same whisper. The specific error function is unknown.

Attack Details. To accurately pinpoint a user location, our approach is to extensively measure the "distance" from different vantage points, and use large-scale statistics to infer user's location. Specifically, our attack exploits a key property of Whisper: servers allow anyone to query the nearby list with arbitrarily self-reported GPS values as input, and impose no rate limits on such queries. This effectively helps us to overcome the limitations (*i.e.* random error, coarse granularity) on the returned distance. First, we can reduce or eliminate per-query noise by taking the average distance across numerous queries from the same observation location. Second, even though the absolute distance is still not accurate, we can estimate the *direction* to the victim based on the measurements from different locations. Then with distance and direction, an attacker can repeat the measurement from a location closer to the victim, thus iteratively deducing the victim's real location.

We use a simple example to illustrate how this works. Suppose user A (attacker) finds user B (victim)'s whisper in the nearby list, and A wants to pinpoint B's location:

1. A queries the nearby list to get its current distance (d) to victim B (averaged across multiple queries).
2. To estimate the direction, A needs additional observation points. We pick 8 points $\{A_1, A_2, ...A_8\}$ evenly distributed on a circle centered at A with radius d (Figure 2.44). From each point, A queries the nearby list to measure its distance to victim $\{d_1, d_2, ..., d_8\}$. Suppose X is a dot on the circle, then objective function $Obj = \sqrt{\frac{\sum_{i=1}^{8}(|\overrightarrow{A_iX}|-d_i)^2}{8}}$ reaches the minimum if $\overrightarrow{AX}$ is the right direction to the victim.
3. Then the attacker moves to the next location using $\overrightarrow{AX}$ and d, and repeats step 1 and 2. The algorithm terminates if $d < Thre_1$, or the distance d from two consecutive rounds differs $< Thre_2$.

In practice, the attacker can script all queries with forged GPS values and does not need to physically move.

Distance Error Correction. Finally, we introduce a final step that uses physical measurements to calibrate and add an additional "correction" factor to location data.

We first post a target whisper at a predefined physical location L (on UCSB campus).

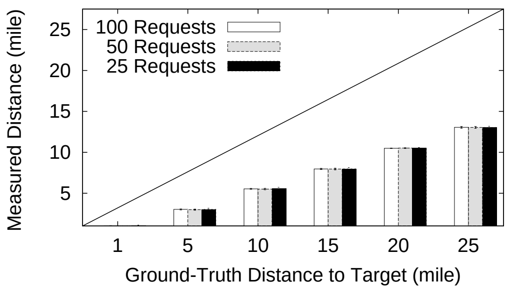


Figure 2.45: True distance vs. measured average distance (>1 mile).

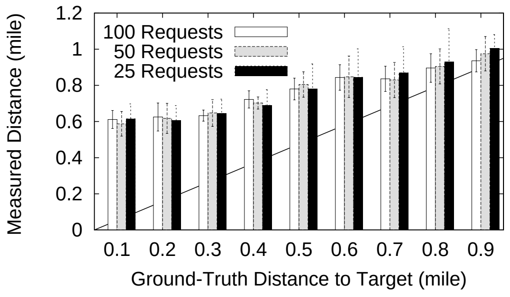


Figure 2.46: True distance vs. measured average distance (within 1 mile).

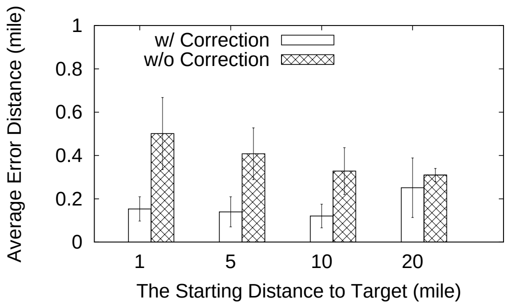


Figure 2.47: The final error distance of the attack.

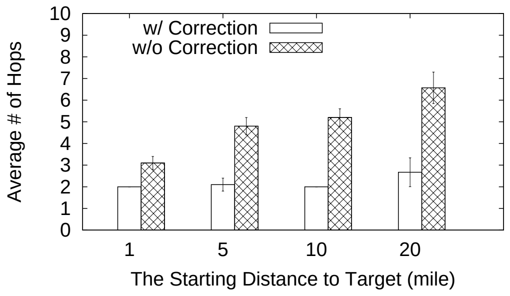


Figure 2.48: Number of hops to approach the victim.

Then we measure distances to L using the nearby list from a set of observation points, each with known ground-truth distances to L. The ground-truth distance ranges cover from 1 to 25 miles (in 5 mile increments) and again from 0.1 to 0.9 miles (in 0.1-mile increments). At each increment, we use 8 observation points (as specified above) and use each to query the nearby list 100 times. Figure 2.45 and Figure 2.46 plot the ground-truth distance versus the measured distance (for 25, 50 and 100 requests per location). For distances greater than 1 mile, we find that our estimates underestimate true physical distance to the victim. Within 1 mile, it clearly overestimates. This mapping between true and measured distance serves as a guide for generating our "correction factor," which is applied to the final estimate.

2.2.7.2 Experimental Validation of the Attack

A Single-target Experiment. We first post a whisper at a pre-defined location on UCSB campus as the target (victim). Then we run the attack algorithm starting from distances of 1, 5, 10 and 20 miles away from the victim. Our algorithm takes the average distance over 50 queries per location, and terminates when the estimated distance from consecutive rounds differ < 0.1 mile or when estimated distance < 0.5 mile (based on Figure 2.46). We repeat each experiment 10 times and test the performance with and without our distance *error correction factor*. Results are shown in Figure 2.47 and Figure 2.48.

We make two key observations. First, the algorithm is very accurate. The final error distance, *i.e.* distance from the estimated victim location to the ground-truth location, is only 0.1 to 0.2 miles. With a radius of 0.2 miles, attackers can already effectively identify user's significant points of interest (*e.g.*, home, work, shopping mall) and reconstruct a victim's daily routine using mobility traces [41]. Second, the results show that distance error correction improves algorithm accuracy significantly and reduces the number of iterations needed to determine the victim's location.

Geographically Diverse Targets. To make sure our results are not biased and specific to a single location, we apply the correction factor computed from local measurements (Figure 2.45 and Figure 2.46) to carry out attacks in different cities. More specifically, we post target whispers in Santa Barbara and Seattle Washington, Denver Colorado, New York City, New York and Edinburgh Scotland. All whispers are posted via an Android phone with forged GPS coordinates. Then we run the algorithm with distance error correction. We find the final error distances are consistently less than 0.2 miles, and that our correction factor can be generalized to improve estimation accuracy regardless of geographic region.

2.2.7.3 Countermeasures

This type of statistical attack cannot be mitigated simply by adding more noise into the system. Attackers can always apply increasingly sophisticated statistical and data mining tools to eliminate noise and determine the true location of a whisper. Instead, the key is to restrict user access to extensive distance measurements. This means putting more constraints (*e.g.*, rate limits) on queries to the nearby list. For instance, one approach is to enforce per-device rate limits. Another is detect fake GPS values, either by relying on client hardware (difficult) or by detecting "unrealistic" movement patterns by potential attackers. Finally, the ultimate defense is to simply remove the "distance" field altogether. While the Whisper engineering team has already addressed this issue, we are not aware of the specific steps they took to do so.

2.2.8 Summary of Results

Anonymous, mobile-only messaging apps such as Whisper mark a clear shift away from traditional social networks and towards privacy-conscious communication tools. To the best of our knowledge, our study is the first large data-driven study of social interactions, user engagement, content moderation and privacy risks on the Whisper network. We show that without strong user identities or persistent social links, users interact with random strangers instead of a defined set of friends, leading to weak ties and challenges in long-term user engagement. We show that even in anonymous messaging apps, significant attacks against user privacy are very feasible. We believe that this shift towards privacy in communication tools is here to stay, and insights from our study on Whisper provides value for developers working on next generation systems in this space.

2.3 Mobility and User Locations.

2.3.1 Introduction

Crowdsourcing is indispensable as a real-time data gathering tool for today's online services. Take for example map and navigation services. Both Google Maps and Waze use periodic GPS readings from mobile devices to infer traffic speed and congestion levels on streets and highways. Waze, the most popular crowdsourced map service, offers users more ways to actively share information on accidents, police cars, and even contribute content like editing roads, landmarks, and local fuel prices. This and the ability to interact with nearby users made Waze extremely popular, with an estimated 50 million users when it was acquired by Google for a reported $1.3 Billion USD in June 2013. Today, Google integrates selected crowdsourced data (*e.g.* accidents) from Waze into its own Maps application.

Unfortunately, systems that rely on crowdsourced data are inherently vulnerable to mischievous or malicious users seeking to disrupt or game the system [203]. For example, business owners can badmouth competitors by falsifying negative reviews on Yelp or TripAdvisor, and FourSquare users can forge their physical locations for discounts [59, 251]. For location-based services, these attacks are possible because there are no widely deployed tools to authenticate the location of mobile devices. In fact, there are few effective tools today to identify whether the origin of traffic requests are real mobile devices or software scripts.

The goal of our work is to explore the vulnerability of today's crowdsourced mobile apps against *Sybil devices*, software scripts that appear to application servers as "virtual mobile devices."[6] While a single Sybil device can damage mobile apps through misbehavior, larger groups of Sybil devices can overwhelm normal users and significantly disrupt any crowdsourced mobile app. In this part of the chapter, we identify techniques that allow malicious attackers to reliably create large populations of Sybil devices using software. Using the context of the Waze crowdsourced map service, we illustrate the powerful Sybil device attack, and then develop and evaluate robust defenses against them.

While our experiments and defenses are designed with Waze (and crowdsourced maps) in mind, our results generalize to a wide range of mobile apps. With minimal modifications, our

[6] We refer to these scripts as Sybil devices, since they are the manifestations of Sybil attacks [71] in the context of mobile networks.

techniques can be applied to services ranging from Foursquare and Yelp to Uber and YikYak, allowing attackers to cheaply emulate numerous virtual devices with forged locations to overwhelm these systems via misbehavior. Misbehavior can range from falsely obtaining coupons on FourSquare/Yelp, gaming the new user coupon system in Uber, to imposing censorship on YikYak. We believe our proposed defenses can be extended to these services as well. We discuss broader implications of our work in Section 2.3.8.

Sybil attacks in Waze. In the context of Waze, our experiments reveal a number of potential attacks by Sybil devices. First is simple *event forgery*, where devices can generate fake events to the Waze server, including congestion, accidents or police activity that might affect user routes. Second, we describe techniques to reverse engineer mobile app APIs, thus allowing attackers to create lightweight scripts that effectively emulate a large number of virtual vehicles that collude under the control of a single attacker. We call Sybil devices in Waze "ghost riders." These Sybils can effectively magnify the efficacy of any attack, and overwhelm contributions from any legitimate users. Finally, we discover a significant privacy attack where ghost riders can silently and invisibly "follow" and precisely track individual Waze users throughout their day, precisely mapping out their movement to work, stores, hotels, gas station, and home. We experimentally confirmed the accuracy of this attack against our own vehicles, quantifying the accuracy of the attack against GPS coordinates. Magnified by an army of ghost riders, an attacker can potentially track the constant whereabouts of millions of users, all without any risk of detection.

Defenses. Prior proposals to address the location authentication problem have limited appeal, because of reliance on widespread deployment of specialized hardware, either as part of physical infrastructure, *i.e.*, cellular base stations, or as modifications to mobile devices themselves. Instead, we propose a practical solution that limits the ability of Sybil devices to amplify the potential damage incurred by any single attacker. We introduce *collocation edges*, authenticated records that attest to the one-time physical proximity of a pair of mobile devices. The creation of collocation edges can be triggered opportunistically by the mapping service, *e.g.*, Waze. Over time, collocation edges combine to form large *proximity graphs,* network structures that attest to physical interactions between devices. Since ghost riders cannot physically interact with real devices, they cannot form direct edges with real devices, only indirectly through a small number of real devices operated by the attacker. Thus, the edges between an attacker and the rest of the network are limited by the number of real physical devices she has, regardless of how many ghost riders are under her control. This reduces the problem of detecting ghost riders to a community detection problem on the proximity graph (The graph is seeded by a small number of trusted infrastructure locations).

We have four key contributions:

- We explore limits and impacts of single device attacks on Waze, *e.g.*, artificial congestion and events.
- We describe techniques to create light-weight ghost riders, virtual vehicles emulated by client-side scripts, through reverse engineering of the Waze app's communication protocol with the server.
- We identify a new privacy attack that allows ghost riders to virtually follow and track individual Waze users in real-time, and describe techniques to produce precise, robust location updates.
- We propose and evaluate defenses against ghost riders, using *proximity graphs* constructed with edges representing authenticated collocation events between pairs of devices. Since collocation can only occur between pairs of physical devices, proximity graphs limit the number of edges between real devices and ghost riders, thus isolating groups of ghost riders and making them detectable using community detection algorithms.

2.3.2 Waze Background

Waze is the most popular crowdsourced navigation app on smartphones, with more than 50 million users when it was acquired by Google in June 2013 [85]. Waze collects GPS values of users' devices to estimate real-time traffic. It also allows users to report on-road events such as accidents, road closures and police vehicles, as well as curating points of interest, editing roads, and even updating local fuel prices. Some features, *e.g.*, user reported accidents, have been integrated into Google Maps [87]. Here, we briefly describe the key functionality in Waze as context for our work.

Trip Navigation. Waze's main feature is assist users to find the best route to their destination and turn-by-turn navigation. Waze generates aggregated real-time traffic updates using GPS data from its users, and optimizes user routes both during trip planning and during navigation. If and when traffic congestions is detected, Waze automatically re-routes users towards an alternative.

Crowdsourced User Reports. Waze users can generate real-time *event reports* on their routes to inform others about ongoing incidents. Events range from accidents to road closures, hazards, and even police speed traps. Each report can include a short note with a photo. The event shows up on the map of users driving towards the reported location. As users get close, Waze pops up a window to let the user "say thanks," or report the event is "not there." If multiple users choose "not there", the event will be removed. Waze also merges multiple reports of the same event type at the same location into a single event.

Social Function. To increase user engagement, Waze supports simple social interactions. Users can see avatars and locations of nearby users. Clicking on a user's avatar shows more detailed user information, including nickname, ranking, and traveling speed. Also, users can send messages and chat with nearby users. This social function gives users the sense of a large community. Users can elevate their rankings in the community by contributing and receiving “thanks” from others.

2.3.3 Attacking Crowdsourced Maps

In this section, we describe basic attacks to manipulate Waze by generating false road events and fake traffic congestion. Since Waze relies on real-time data for trip planning and route selection, these attacks can influence user’s routing decisions. Attackers can attack specific users by forging congestion to force automatic rerouting on their trips. The attack is possible because Waze has no reliable authentication on user reported data, such as their device GPS.

We first discuss experimental ethics and steps we took to limit impact on real users. Then, we describe basic mechanisms and resources needed to launch attacks, and use controlled experiments on two attacks to understand their feasibility and limits. One attack creates fake road events at arbitrary locations, and the other seeks to generate artificial traffic hotspots to influence user routing.

2.3.3.1 Ethics

Our experiments seek to understand the feasibility and limits of practical attacks on crowdsourcing maps like Waze. We are very aware of the potential impact to real Waze users from any experiments. We consulted our local IRB and have taken all possible precautions to ensure that our experiments do not negatively impact real Waze users. In particular, we choose experiment locations where user population density is extremely low (unoccupied roads), and only perform experiments at low-traffic hours, *e.g.*, between 2am and 5am. During the experiments, we continuously scan the entire experiment region and neighboring areas, to ensure no other Waze users (except our own accounts) are within miles of the test area. If any Waze users are detected, we immediately terminate all running experiments. Our study received the IRB approval under protocol# COMS-ZH-YA-010-7N.

Our work is further motivated by our view of the risks of inaction versus risks posed to users by our study. On one hand, we can and have minimized risk to Waze users during our study, and we believe our experiments have not affected any Waze users. On the other hand, we believe the risk to millions of Waze users from pervasive location tracking (described in Section 2.3.5) is realistic and potentially very damaging. We feel that investigating these attacks and identifying these risks to the broad community at large was the ethically correct course of action. Furthermore, full understanding of the attacks was necessary to design an effective and *practical* defense. Please see Appendix A for more detailed information on our IRB approval and steps taken towards responsible disclosure.

2.3.3.2 Basic Attack: Generating Fake Events

Launching attacks against crowdsourced maps like Waze requires three steps: automate input to mobile devices that run the Waze app; control the device GPS and simulate device movements (*e.g.,* car driving); obtain access to *multiple* devices. All three are easily achieved using widely available mobile device emulators.

Most mobile emulators run a full OS (*e.g.,* Android, iOS) down to the kernel level, and simulate hardware features such as camera, SDCard and GPS. We choose the GenyMotion Android emulator [2] for its performance and reliability. Attackers can automatically control the GenyMotion emulator via Monkeyrunner scripts [3]. They can generate user actions such as clicking buttons and typing text, and feed pre-designed GPS sequences to the emulator (through a command line interface) to simulate location positioning and device movement. By controlling the timing of the GPS updates, they can simulate any "movement speed" of the simulated devices.

Using these tools, attackers can generate fake events (or alerts) at a given location by setting fake GPS on their virtual devices. This includes any events supported by Waze, including

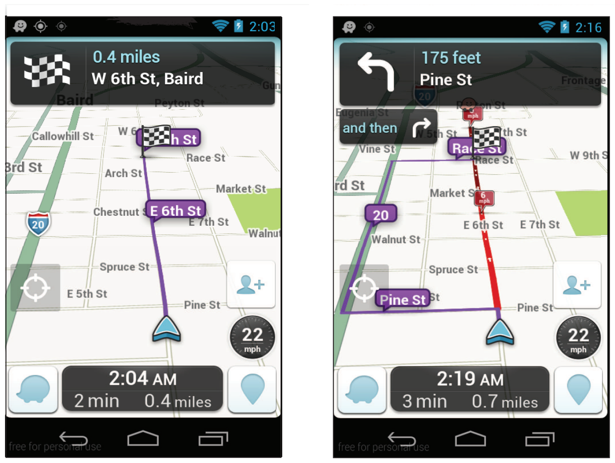


Figure 2.49: Before the attack (left), Waze shows the fastest route for the user. After the attack (right), the user gets automatically re-routed by the fake traffic jam.

accidents, police, hazards, and road closures. We find that a single emulator can generate any event at arbitrary locations on the map. We validate this using experiments on a variety of unoccupied roads, including highways, local and rural roads (50+ locations, 3 repeated tests each). Note that our experiments only involve data in the Waze system, and do not affect real road vehicles not running the Waze app. Thus “unoccupied” means no vehicles on the road with mobile devices actively running the Waze app. After creation, the fake event stays on the map for about 30 minutes. Any Waze user can report that an event was “not there.” We find it takes two consecutive “not theres” (without any “thanks” in between) to delete the event. Thus an attacker can ensure an event persists by occasionally “driving” other virtual devices to the region and “thanking” the original attacker for the event report.

2.3.3.3 Congestion and Traffic Routing

A more serious attack targets Waze’s real-time trip routing function. Since route selection in Waze relies on predicted trip time, attackers can influence routes by creating “fake” traffic hotspots at specific locations. This can be done by configuring a group of virtual vehicles to travel slowly on a chosen road segment.

We use controlled experiments to answer two questions. First, under what conditions can attackers successfully create traffic hotspots? Second, how long can an artificial traffic hotspot last? We select three low-traffic roads in the state of Texas that are representative of three popular road types based on their speed limit—Highway (65 mph), Local (45 mph) and Residential (25 mph). To avoid real users, we choose roads in low population rural areas, and run tests at hours with the lowest traffic volumes (usually 3-5AM). We constantly scan for real users in or nearby the experimental region, and reset/terminate experiments if users come close to an area with ongoing experiments. Across all our experiments, only 2 tests were terminated due to detected presence of real users nearby. Finally, we have examined different road types and hours of the day to ensure they do not introduce bias into our results.

Creating Traffic Hotspots. Our experiment shows that it only takes one slow moving car to create a traffic congestion, when there are no real Waze users around. Waze displays a red overlay on the road to indicate traffic congestion (Figure 2.49, right). Different road types have different congestion thresholds, with thresholds strongly correlated to the speed limit. The congestion thresholds for Highway, Local and Residential roads are 40mph, 20mph and 15mph, respectively.

To understand if this is generalizable, we repeat our tests on other unoccupied roads in different states and countries. We picked 18 roads in five states in the US (CO, MO, NM, UT, MS) and British Columbia, Canada. In each region, we select three roads with different speed limits (highway, local and residential). We find consistent results: a single virtual vehicle can always generate a traffic hotspot; and the congestion thresholds were consistent across different roads of the same speed limit.

Outvoting Real Users. Generating traffic hotspot in practical scenarios faces a challenge from real Waze users who drive at normal (non-congested) speeds: attacker's virtual vehicles must "convince" the server there's a stream of slow speed traffic on the road even as real users

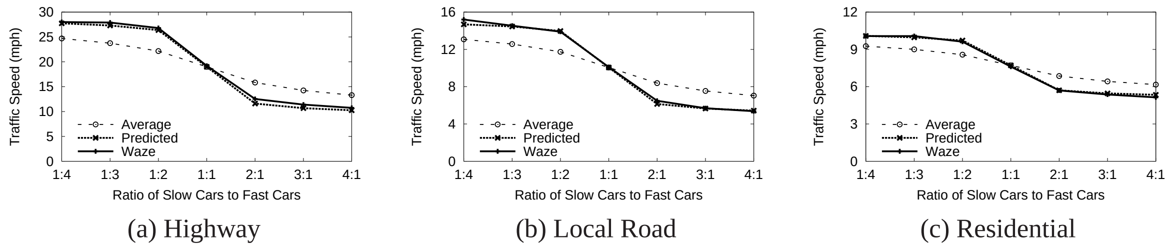


Figure 2.50: The traffic speed of the road with respect to different combinations of number of slow cars and fast cars. We show that Waze is not using the average speed of all cars, and our inferred function can correctly predict the traffic speed displayed on Waze.

tell the server otherwise. We need to understand how Waze aggregated multiple inputs to estimate traffic speed.

We perform an experiment to infer this aggregation function used by Waze. We create two groups of virtual vehicles: N_s slow-driving cars with speed S_s, and N_f fast-driving cars with speed S_f; and they all pass the target location at the same time. We study the congestion reported by Waze to infer the aggregation function. Note that the server-estimated traffic speed is visible on the map *only if* we formed a traffic hotspot. We achieve this by setting the speed tuple (S_s, S_f) to (10mph, 30mph) for Highway, (5, 15) for Local and (5, 10) for Residential.

As shown in Figure 2.50, when we vary the ratio of slow cars over fast cars (N_s:N_f), the Waze server produces different final traffic speeds. We observe that Waze does not simply compute an “average” speed over all the cars. Instead, it uses a weighted average with higher weight on the majority cars’ speed. We infer an aggregation function as follows:

$$S_{waze} = \frac{S_{max} \cdot max(N_s, N_f) + S_{avg} \cdot min(N_s, N_f)}{N_s + N_f}$$

where $S_{avg} = \frac{S_s N_s + S_f N_f}{N_s + N_f}$, and S_{max} is the speed of the group with N_{max} cars. As shown in Figure 2.50, our function can predict Waze’s aggregate traffic speed accurately, for all different types of roads in our test. For validation purposes, we run another set of experiments by raising S_f above the hotspot thresholds (65mph, 30mph and 20mph respectively for the three roads).

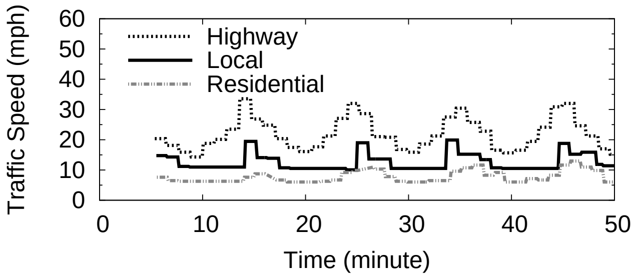


Figure 2.51: Long-last traffic jam created by slow cars driving-by.

We can still form traffic hotspots by using more slow-driving cars ($N_s > N_f$), and our function can still predict the traffic speed on Waze accurately.

Long-Lasting Traffic Congestion. A traffic hotspot will last for 25-30 minutes if no other cars drive by. Once aggregate speed normalizes, the congestion event is dismissed within 2-5 minutes. To create a long-lasting virtual traffic jam, attackers can simply keep sending slow-driving cars to the congestion area to resist the input from real users. We validate this using a simple, 50-minute long experiment where 3 virtual vehicles create a persistent congestion by driving slowly through an area, and then looping back every 10 minutes. Meanwhile, 2 other virtual cars emulate legitimate drivers that pass by at high speed every 10 minutes. As shown in Figure 2.51, the traffic hotspot persists for the entire experiment period.

Impact on End Users. Waze uses real-time traffic data to optimize routes during trip planning. Waze estimates the end-to-end trip time and recommends the fastest route. Once on the road, Waze continuously estimates the travel time, and automatically reroutes if the current route becomes congested. An attacker can launch physical attacks by placing fake traffic hotspots on the user's original route. While congestion alone does not trigger rerouting, Waze reroutes the user to a detour when the estimated travel time through the detour is shorter than the current congested route (see Figure 2.49).

We also note that Waze data is used by Google Maps, and therefore can potentially impact their 1+ billion users [181]. Our experiment shows that artificial congestion do not appear on

Google Maps, but fake events generated on Waze are displayed on Google Maps without verification, including “accidents”, “construction” and “objects on road”. Finally, event updates are synchronized on both services, with a 2-minute delay and persist for a similar period of time (*e.g.*, 30 minutes).

2.3.4 Sybil Attacks

So far, we have shown that attackers using emulators can create “virtual vehicles” that manipulate the Waze map. An attacker can generate much higher impact using a large group of virtual vehicles (or *Sybils* [71]) under control. In this section, we describe techniques to produce light-weight virtual vehicles in Waze, and explore the scalability of the group-based attacks. We refer to large groups of virtual vehicles as “ghost riders” for two reasons. First, they are easy to create en masse, and can travel in packs to outvote real users to generate more complex events, *e.g.*, persistent traffic congestion. Second, as we show in §2.3.5, they can make themselves invisible to nearby vehicles.

Factors Limiting Sybil Creation. We start by looking at the limits of the large-scale Sybil attacks on Waze. First, we note user accounts do not pose a challenge to attackers, since account registration can be fully automated. We found that a single-threaded Monkeyrunner script could automatically register 1000 new accounts in a day. Even though the latest version of Waze app requires SMS verification to register accounts, attackers can use older versions of APIs to create accounts without verification. Alternatively, accounts can be verified through disposable phone/SMS services [214].

The limiting factor is the scalability of vehicle emulation. Even though emulators like GenyMotion are relatively lightweight, each instance still takes significant computational resources. For example, a MacBookPro with 8G of RAM supports only 10 simultaneous emulator instances. For this, we explore a more scalable approach to client emulation that can

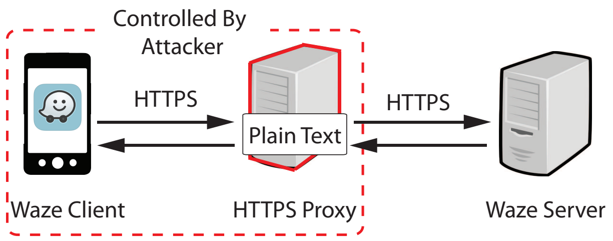


Figure 2.52: Using a HTTPS proxy as man-in-the-middle to intercept traffic between Waze client and server.

increase the number of supported virtual vehicles by orders of magnitude. Specifically, we reverse engineer the communication APIs used by the app, and replace emulators with simple Python scripts that mimic API calls.

Reverse Engineering Waze APIs. The Waze app uses HTTPS to communicate with the server, so API details cannot be directly observed by capturing network traffic (TLS/SSL encrypted). However, an attacker can still intercept HTTPS traffic, by setting up a proxy [1] between her phone and Waze server as a man-in-the-middle attack [199, 57]. As shown in Figure 2.52, an attacker needs to pre-install the proxy server's root Certificate Authorities (CA) to her own phone as a "trusted CA." This allows the proxy to present self-signed certificates to the phone claiming to be the Waze server. The Waze app on the phone will trust the proxy (since the certificate is signed by a "trusted CA"), and establish HTTPS connections with the proxy using proxy's public key. On the proxy side, the attacker can decrypt the traffic using proxy's private key, and then forward traffic from the phone to Waze server through a separate TLS/SSL channel. The proxy then observes traffic to the Waze servers and extracts the API calls from plain text traffic.

Hiding API calls using traffic encryption is fundamentally challenging, because the attacker has control over most of the components in the communication process, including phone, the app binary, and the proxy. A known countermeasure is certificate pinning [75], which embeds a copy of the server certificate within the app. When the app makes HTTPS requests, it validates the server-provided certificate with its known copy before establishing connections. However, dedicated attackers can extract and replace the embedded certificate by disassembling the app binary or attaching the app to a debugger [168, 74].

Scalability of Ghost Riders. With the knowledge of Waze APIs, we build extremely lightweight Waze clients using python scripts, allocating one thread for each client. Within each thread, we log in to the app using a separate account, and maintain a live session by sending periodic GPS coordinates to the Waze server. The Python client is a full Waze client, and can report fake events using the API. Scripted emulation is highly scalable. We run 1000 virtual vehicles on a single Linux Dell PowerEdge Server (Quad Core, 2GB RAM), and find that at steady state, 1000 virtual devices only introduces a small overhead: 11% of memory usage, 2% of CPU and 420 Kbps bandwidth. In practice, attackers can easily run tens of thousands of virtual devices on a commodity server.

Finally, we experimentally confirm the practical efficacy and scalability of ghost riders. We chose a secluded highway in rural Texas, and used 1000 virtual vehicles (hosted on a single server and single IP) to generate a highly congested traffic hotspot. We perform our experiment in the middle of the night after repeated scans showed no Waze users within miles of our test area. We positioned 1000 ghost riders one after another, and drove them slowly at 15 mph along the highway, looping them back every 15 minutes for an entire hour. The congestion shows up on Waze 5 minutes after our test began, and stayed on the map during the entire test period. No problems were observed during our test, and tests to generate fake events (accidents etc.) also succeeded.

2.3.5 User Tracking Attack

Next, we describe a powerful new attack on user privacy, where virtual vehicles can track Waze users continuously without risking detection themselves. By exploiting a key social functionality in Waze, attackers can remotely follow (or stalk) any individual user in real time.

This is possible with single device emulation, but greatly amplified with the help of large groups of ghost riders, possibly tracking large user populations simultaneously and putting user (location) privacy at great risk. We start by examining the feasibility (and key enablers) of this attack. We then present a simple but highly effective tracking algorithm that follows individual users in real time, which we have validated using real life experiments (with ourselves as the targets).

The only way for Waze users to avoid tracking is to go "invisible" in Waze. However, doing so forfeits the ability to generate reports or message other users. Users are also reset to "visible" each time the Waze app opens.

2.3.5.1 Feasibility of User Tracking

A key feature in Waze allows users to socialize with others on the road. Each user sees on her screen icons representing the locations of nearby users, and can chat or message with them through the app. Leveraging this feature, an attacker can pinpoint any target who has the Waze app running on her phone. By constantly "refreshing" the app screen (issuing an update query to the server), an attacker can query the victim's GPS location from Waze in real time. To understand this capability, we perform detailed measurements on Waze to evaluate the efficiency and precision of user tracking.

Tracking via User Queries. A Waze client periodically requests updates in her nearby area, by issuing an update query with its GPS coordinates and a rectangular "search area." This search area can be set to any location on the map, and does not depend on the requester's own location. The server returns a list of users located in the area, including userID, nickname, account creation time, GPS coordinates and the GPS timestamp. Thus an attacker can find and "follow" a target user by first locating them at any given location (work, home) and then continuously following them by issuing update queries centered on the target vehicle location,

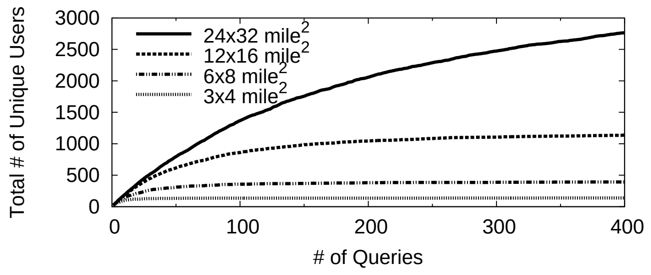


Figure 2.53: # of queries vs. unique returned users in the area.

all automated by scripts.

Overcoming Downsampling. The user query approach faces a downsampling challenge, because Waze responds to each query with an "incomplete" set of users, *i.e.*, up to 20 users per query regardless of the search area size. This downsampled result is necessary to prevent flooding the app screen with too many user icons, but it also limits an attacker's ability to follow a moving target.

This downsampling can be overcome by simply repeatedly querying the system until the target is found. We perform query measurements on four test areas (of different sizes between 3×4 mile^2 and 24×32 mile^2) in the downtown area of Los Angeles (City A, with 10 million residents as of 2015). For each area, we issue 400 queries within 10 seconds, and examine the number of unique users returned by all the queries. Results in Figure 2.53 show that the number of unique users reported converges after 150-250 queries for the three small search areas ($\leq 12 \times 16$ mile^2). For the area of size 24×32 mile^2, more than 400 queries are required to reach convergence.

We confirm this "downsampling" is uniformly random, by comparing our measurement results to a mathematical model that projects the statistics of query results assuming uniform-random sampling. Consider total M users in the search area. The probability of a user x getting sampled in a single round of query (20 users per query) is $P(x) = \frac{20}{M}$. Over N queries, the number of appearances per user should follow a Binomial Distribution [119] with mean $N \cdot \frac{20}{M}$. Figure 2.54 plots the measured user appearances for the four servers on the 6×8 mile^2 area with

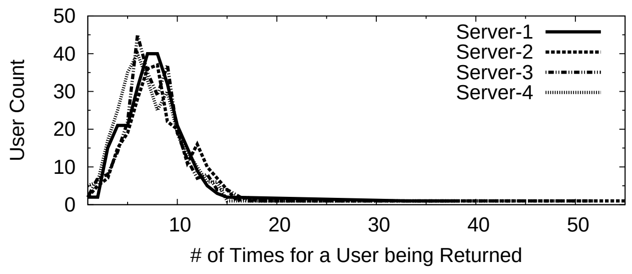


Figure 2.54: User's number of appearances in the returned results (6×8 mile^2 area).

$N = 100$. The measured statistics follow the projected Binomial Distribution (the measured mean values closely match the theoretical expectation). This confirms that the downsampling is indeed random, and thus an attacker can recover a (near) complete set of Waze users with repeated queries. While the number of queries required increases superlinearly with area size, a complementary technique is to divide an area into smaller, fixed size partitions and query each partition's users in parallel.

We also observe that user lists returned by different Waze servers had only a partial overlap (roughly 20% of users from each server were unique to that server). This "inconsistency" across servers is caused by synchronization delay among the servers. Each user only sends its GPS coordinates to a single server which takes 2-5 minutes to propagate to other servers. Therefore, a complete user set requires queries to cover all Waze servers. At the time of our experiments, the number of Waze servers could be traced through app traffic and could be covered by a moderate number of querying accounts.

Tracking Users over Time. Our analysis found that each active Waze app updates its GPS coordinates to the server every 2 minutes, regardless of whether the user is mobile or stationary. Even when running in the background, the Waze app reports GPS values every 5 minutes. As long as the Waze app is open (even running in the background), the user's location is continuously reported to Waze and potential attackers. Clearly, a more conservative approach to managing location data would be extremely helpful here.

We note that attackers can perform long-term tracking on a target user (*e.g.*, over months). The attacker needs a persistent ID associated to the target. The "userID" field in the metadata is insufficient, because it is a random "session" ID assigned upon user login and is released when the user kills the app. However, the "account creation time" can serve as a persistent ID, because a) it remains the same across the user's different login sessions, and b) it is precise down to the second, and is sufficiently to uniquely identify single users in the same geographic area. While Waze can remove the "account creation time" field from metadata, a persistent attacker can overcome this by analyzing the victim's mobility pattern. For example, the attacker can identify a set of locations where the victim has visited frequently or stayed during the past session, mapping to home or workplace. Then the attacker can assign a ghost rider to constantly monitor those areas, and re-identify the target once her icon shows up in a monitored location, *e.g.*, home.

Stealth Mode. We note that attackers remain invisible to their targets, because queries on any specific geographic area can be done by Sybils operating "remotely," i.e. claiming to be in a different city, state or country. Attackers can enable their "invisible" option to hide from other nearby users. Finally, disabling these features still does not make the attacker visible. Waze only updates each user's "nearby" screen every 2 minutes (while sending its own GPS update to the servers). Thus a tracker can "pop into" the target's region, query for the target, and then move out of the target's observable range, all before the target can update and detect it.

2.3.5.2 Real-time Individual User Tracking

To build a detailed trace of a target user's movements, an attacker first bootstraps by identifying the target's icon on the map. This can be done by identifying the target's icon while confirming her physical presence at a time and location. The attacker centers its search area

Location	Route Len. (mile)	Travel Time (min)	GPS Sent By Victim	GPS Captured by Attacker	To Destination?	Track Delay (sec.)	User Density (# Users/mile^2)
City A	12.8	35	18	16	Yes	43.79	56.6
Highway B	36.6	40	20	19	Yes	9.24	2.8

Table 2.7: Tracking Experiment Results.

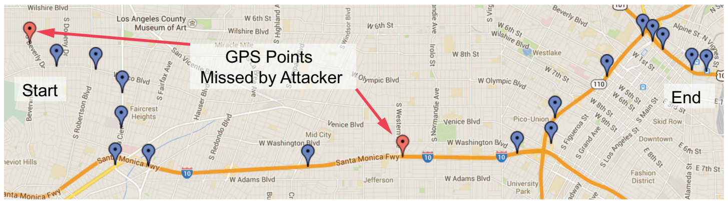


Figure 2.55: A graphical view of the tracking result in Los Angeles downtown (City A). Blue dots are GPS points captured by the attacker and the red dots are those missed by the attacker.

on the victim's location, and issues a large number of queries (using Sybil accounts) until it captures the next GPS report from the target. If the target is moving, the attacker moves the search area along the target's direction of movement and repeats the process to get updates.

Experiments. To evaluate its effectiveness, we performed experiments by tracking one of our own Android smartphones and one of our virtual devices. Tracking was effective in both cases, but we experimented more with tracking our virtual device, since we could have it travel to any location. Using the OSRM tool [4], we generate detailed GPS traces of two driving trips, one in downtown area of Los Angeles (City A), and one along the interstate highway-101 (Highway B). The target device uses a realistic driving speed based on average traffic speeds estimated by Google Maps during the experiment. The attacker used 20 virtual devices to query Waze simultaneously in a rectangular search area of size 6×8 mile^2. This should be sufficient to track the GPS update of a fast-driving car (up to 160 mph). Both experiments were during morning hours, and we logged both the network traffic of the target phone and query data retrieved by the attacker. Note that we did not generate any "events" or otherwise affect the Waze system in this experiment.

Results. Table 2.7 lists the results of tracking our virtual device, and Figure 2.55 presents a graphical view of the City A result. For both routes, the attacker can consistently follow the victim to her destination, though the attacker fails to capture 1-2 GPS points out of the 18-20 reported. For City A, the tracking delay, *i.e.*, the time spent to capture the subsequent GPS of the victim, is larger (averaging 43s rather than 9s). This is because the downtown area has a higher Waze user density, and required more rounds of queries to locate the target.

Our experiments represent two highly challenging (*i.e.*, worst case) scenarios for the attacker. The high density of Waze users in City A downtown is makes it challenging to locate a target in real time with downsampling. On Highway B, the target travels at a high speed (∼60mph), putting a stringent time limit on the tracking latency, *i.e.*, the attacker must capture the target before he leaves the search area. The success of both experiments confirms the effectiveness and practicality of the proposed attack.

2.3.6 Defenses

In this section, we discuss potential defense mechanisms to limit the magnitude and impact of these attacks. While individual devices can inflict limited damage, an attacker's ability to control a large number of virtual vehicles at low cost elevates the severity of the attack in both quantity and quality. Our priority, then, is to restrict the number of ghost riders available to each attacker, thus increasing the cost per "vehicle" and reducing potential damage.

The most intuitive approach is perform strong location authentication, so that attackers must use real devices physically located at the actual locations reported. This would make ghost riders as expensive to operate as real devices. Unfortunately, existing methods for location authentication do not extend well to our context. Some proposals solely rely on trusted infrastructures (*e.g.*, wireless access points) to verify the physical presence of devices in close proximity [140, 191]. However, this requires large scale retrofitting of cellular celltowers or installation of new hardware, neither of which is practical at large geographic scales. Others propose to embed tamperproof location hardware on mobile devices [145, 192], which incurs high cost per user, and is only effective if enforced across all devices. For our purposes, we need a scalable approach that works with current hardware, without incurring costs on mobile users or the map service (Waze). In the following, we briefly describe the high level idea of proposed defense and direct interested readers to our full paper for the detailed system design and evaluation [228].

2.3.6.1 Sybil Detection via Proximity Graph

Instead of optimizing per-device location authentication, our proposed defense is a Sybil detection mechanism based on the novel concept of *proximity graph*. Specifically, we leverage physical proximity between real devices to create *collocation edges*, which act as secure attestations of shared physical presence. In a proximity graph, nodes are Waze devices (uniquely identified by an account username and password on the server side). They perform secure peer-to-peer location authentication with the Waze app running in the background. An edge is established if the proximity authentication is successful.

Because Sybil devices are scripted software, they are highly unlikely to come into physical proximity with real devices. A Sybil device can only form collocation edges with other Sybil devices (with coordination by the attacker) or the attacker's own physical devices. The resulting graph should have only very few (or no) edges between virtual devices and real users (other than the attacker). Leveraging prior work on Sybil detection in social networks, groups of Sybils can be characterized by the few "attack edges" connecting them to the rest of the graph, making them identifiable through community-detection algorithms [221].

We use *a very small number* of trusted nodes only to bootstrap trust in the graph. We assume a small number of infrastructure access points are known to Waze servers, *e.g.*, hotels and public WiFi networks associated with physical locations stored in IP-location databases (used for geolocation by Apple and Google). Waze also can work with merchants that own public WiFi access points (*e.g.*, Starbucks). These infrastructures are trusted nodes (we assume trusted nodes don't collude with attackers). Any Waze device that communicates with the Waze server under their IPs (and reports a GPS location consistent with the IP) automatically creates a new collocation edge to the trusted node.

2.3.6.2 Alternative Defenses

In addition to Sybil detection, Waze can incorporate other mechanisms to protect its users. We briefly describe a few key ideas, but leave the integration with our approach to future work. *First*, IP verification: when a user claims she is driving, Waze can examine whether her IP is a mobile IP that belongs to a valid cellular carrier or a suspicious web proxy. However, this approach is ineffective if dedicated attackers route the attack traffic through a cellular data plan. *Second*, strict rate limit: with that, attackers will need to run more Sybil devices to implement the same attack. *Third*, verifications on account registration: this needs to be handled carefully since email/SMS based verification can be bypassed using disposable email or phone numbers [214]. *Finally*, detecting extremely inconsistent GPS/event reports. The challenge, however, is to distinguish honest reports from the fake ones since attacker can easily outvote real users. If Waze chooses to ignore all the inconsistent reports, it will lead to DOS attack where attackers disable the service with inconsistent data.

2.3.7 Our Interactions with Waze

After our study, we have taken active steps to inform the Google/Waze team of our results and help them to mitigate the threat. In this section, we want to share our experience of interacting with Waze team, and discuss the security measures from Waze and their effectiveness.

Informing Waze Team Directly. Before the first writeup of our work in November 2014, we sought to inform the Google Waze team of our findings. We first used multiple existing Google contacts on the Security and Android teams. When that failed to reach the Waze team, we got in touch with Niels Provos, who then relayed information about our project to the Waze team.

As of October 2015, we observed a major change in Waze app on how the app reports user location data to the server (and other users). In the new version, the app only reports user GPS values when the user is actively driving (moving at a moderate/fast rate of speed). GPS tracking stops when a user is walking or standing still. In addition, Waze automatically shuts down if the user puts it in the background, and has not driven for a while. To resume user tracking (GPS reporting), users must manually bring the app to the foreground. Finally, Waze now hide users' starting and destination locations of their trips. While online documentation claims that these optimizations are to reduce energy usage for the Waze app, we are gratified by the dramatic steps taken to limit user tracking and improve user privacy. These changes indeed reduce the amount of GPS data sent to the server (and made available to potential attackers through the API). By our estimates, the update reduces the amount of GPS tracking data for a typical user by nearly a factor of 10x. However, an attacker can still build Sybil devices to track active Waze users.

Informing Waze via News Media. To further raise awareness on the threat of Sybil devices, in April 2016, we pitched our work to Fusion. We demonstrated the effectiveness of the tracking attack by tracking one of their reporters with her consent for three days. On April 26, 2016, Fusion covered our story, which went viral within 24 hours with followup stories from 20+ media outlets all around the world. This time, Waze immediately issued a response on the next day [5], followed with a series of updates to the app. First, Waze disabled the social feature in older versions (v3.8 or lower). In addition, the latest app uses special encoding on the communication APIs (binary format, no longer human-readable). In the meantime, we tested the app and found that Waze was using Google Protocol Buffer to perform the encoding. We managed to crack the encoding scheme within a day, and validated that our attack still worked. We notified Waze about our findings.

Working with Waze. As of May 2016, the product manager of Waze reached out to us to start a collaboration to improve Waze security. We helped the Waze team to understand our attacking method, and helped to test any security measures they deployed. As a starting point, Waze removed the globally unique identifier of users (account creation time) and username, making it hard to track users over multiple trips. In addition, Waze started to require a two-factor authentication through SMS before showing any identifiable information to nearby users.

To assess the effectiveness of the security measures, we tested our tracking attack on the new app. To bypass the two-factor authentication, we tried to use temporal SMS services (or disposable phone number) [214] to verify the fake accounts. We found that once the account got verified, our Sybil device can communicate with Waze server and the tracking attack still worked. We reported our findings and also suggested other potential security measures such as enforcing a rate limit for queries per device, checking whether the device is using a mobile IP, and detecting unrealistic movement patterns of a device. It is clear that current countermeasure is not perfect and it is an on-going effort to further raise the bar for attackers.

Thus far, our efforts have led to significant improvement of the security and privacy in Waze. After the back-and-forth interaction, much less amount of location information is shared about users. Currently, only active users (who are driving on the road with Waze app on the foreground) can be tracked. It is also much more difficult than before to track users across multiple trips.

2.3.8 Broader Implications

While our experiments and defenses have focused strictly on Waze, our results are applicable to a wider range of mobile applications that rely on geolocation for user-contributed content and metadata. Examples include location based check-in and review services (Foursquare, Yelp), crowdsourced navigation systems (Waze, Moovit), crowdsourced taxi services (Uber, Lyft), mobile dating apps (Tinder, Bumble), anonymous mobile communities (Yik Yak, Whisper) and location-based gaming apps (Pokemon Go).

These systems face two common challenges exposing them to potential attacks. First, our efforts show that it is difficult for app developers to build a truly secure channel between the app and the server. There are numerous avenues for an attacker to reverse-engineer and mimic an app's API calls, thereby creating "cheap" virtual devices and launching Sybil attack [71]. Second, there are no deployed mechanisms to authenticate location data (*e.g.*, GPS report). Without a secure channel to the server and authenticated location, these mobile apps are vulnerable to automated attacks ranging from nuisance (prank calls to Uber) to malicious content attacks (large-scale rating manipulation on Yelp).

Attacking other Apps. To validate our point, we run a quick empirical analysis on a broad class of mobile apps to understand how easy it is to reverse-engineer their APIs and inject falsified data into the system. We pick one app from each category including Foursquare, Uber, Tinder, Yik Yak and Pokemon Go (an incomplete list). We find that, although all the listed apps use TLS/SSL to encrypt their network traffic, their APIs can be fully exposed by the method in §2.3.4. For each app, we were able to build a light-weight client using python script, and feed arbitrary GPS to their key function calls. For example, with forged GPS, a group of Foursquare clients can deliver large volumes of check-ins to a given venue without physically visiting it; On Uber, one can distribute many virtual devices as sensors, and passively monitor and track all drivers within a large area (see §2.3.5). Similarly for Yik Yak and Tinder, the virtual devices make it possible to perform wardriving in a given location area to post and collect anonymous Yik Yak messages or Tinder profiles. In addition, apps like Tinder also display the geographical distance to a nearby user (*e.g.*, 1 mile). Attacker can use multiple virtual devices to measure the distance to the target user, and "triangulate" that user's exact location [227]. Finally, for Pokemon Go, we can use simulated devices to capture pokemons without physically walking outside like other players (cheating in the game). There are possible app-specific defenses, and we leave their design and evaluation to future work.

New Countermeasures in the Wild. After our initial report was published, we start to observe some interesting countermeasures against API reverse-engineerings in these apps. For example, Yik Yak uses an authentication method in their APIs called HMAC (keyed-hash message authentication code). The app developer has embed a key in the app binary, and uses the key to generate the authentication code. API calls without the authentication code are no longer accepted by the server. To build a Sybil device for Yik Yak, the attacker need to take extra effort to extract the key from the app binary. In addition, we observe apps like Twitter, Periscope have adopted SSL pinning, so that the app no longer accept self-signed certificate. This makes it more difficult to set up a HTTPS proxy to learn the API calls. Attacker will need to replace the pinned certificate from the app binary in order to reverse-engineer the API calls as §2.3.4. We believe further research is needed to empirically understand the usage and effectiveness of different countermeasures within a wide range of mobile apps.

2.3.9 Summary of Results

In summary, we describe our efforts to identify and study a range of attacks on crowd-sourced map services. We identify a range of single and multi-user attacks, and describe techniques to build and control groups of virtual vehicles (ghost riders) to amplify these attacks. Our work shows that today's mapping services are highly vulnerable to software agents controlled by malicious users, and both the stability of these services and the privacy of millions of users are at stake. We propose and validate a suite of techniques that help services to build proximity graphs and use them to effectively detect Sybil devices.

While our study and experiments focus on the Waze system, we believe the large majority of our results can be generalized to crowdsourced apps as a group. Broadly speaking, for any apps that support "human-to-human" interactions, they inevitably have to leak some user data (*e.g.*, location data, user identity information) to other users through such interactions. This become a real concern when an attacker controls a large group of Sybil devices to massively interact with human users and retrive or pollute user data.

Chapter 3

Spam, Human Factors and Malicious Crowdsourcing

Thus far, we have discussed the critical challenges in online communities in Chapter 2 regarding quality of content, user anonymity and location privacy. In this chapter, we specifically focus on the generation and distribution of malicious content (*e.g.* spam) in online communities and practical defense techniques. While traditional spam attacks are mostly generated by automated software, more sophisticated attackers today start to introduce "human intelligence" to their attacking process. Through extensive measurements, we find strong evidence on the rising of *malicious crowdsourcing* services where a large number of real users are hired for pennies to perform malicious activities, which poses a significant challenge to existing security systems (*e.g.*, CAPTCHA), which are initially designed to detect attacks from automated software, but become ineffective to real users. In the following, we describe our data-driven approach to understanding and defending against malicious crowdsourcing (or *crowdturfing*).

3.1 Understanding Malicious Crowdsourcing

3.1.1 Introduction

Popular Internet services in recent years have shown that remarkable things can be achieved by harnessing the power of the masses. By distributing tasks or questions to large numbers of Internet users, these "crowd-sourcing" systems have done everything from answering user questions (Quora), to translating books, creating 3-D photo tours [197], and predicting the behavior of stock markets and movie grosses. Online services like Amazon's Mechanical Turk, Rent-a-Coder (vWorker), Freelancer, and Innocentive have created open platforms to connect people with jobs and workers willing to perform them for various levels of compensation.

On the other hand, crowd-sourcing systems could pose a serious challenge to a number of security mechanisms deployed to protect Internet services against automated scripts. For example, electronic marketplaces want to prevent scripts from automating auction bids [146], and online social networks (OSNs) want to detect and remove fake users (Sybils) that spread spam [213, 244]. Detection techniques include different types of CAPTCHAs, as well as machine-learning that tries to detect abnormal user behavior [84], *e.g.* near-instantaneous responses to messages or highly bursty user events. Regardless of the specific technique used, they rely on a common assumption, that the malicious tasks in question cannot be performed by real humans en masse. This is an assumption that is easily broken by crowd-sourcing systems dedicated to organizing works to perform malicious tasks.

Through measurements, we have found surprising evidence showing that not only do malicious crowd-sourcing systems exist, but they are rapidly growing in both user base and revenue generated. Because of their similarity with both traditional crowd-sourcing systems and astroturfing behavior, we refer to them as *crowdturfing* systems. More specifically, we define crowdturfing systems as systems where customers initiate "campaigns," and a significant number of users obtain financial compensation in exchange for performing simple "tasks" that go against accepted user policies.

In this part of the chapter, we describe a significant effort to study and understand crowdturfing systems in today's Internet. We found significant evidence of these systems in a number of countries, including the US and India, but focus our study on two of the largest crowdturfing systems with readily available data, both of which are hosted in and targeted users in China. From anecdotal evidence, we learn that these systems are well-known to young Internet users in China, and have persisted despite threats from law enforcement agencies to shut them down [72, 79, 130].

Our study results in four key findings on the operation and effectiveness of crowdturfing systems. First, we used detailed crawls to extract data about the size and operational structure of these crowdturfing systems. We use readily available data to quantify both tasks and revenue flowing through these systems, and observe that these sites are growing exponentially in both metrics. Second, we study the types of tasks offered and performed in these sites, which include mass account creation, and posting of specific content on OSNs, microblogs, blogs, and online forums. Tasks often ask users to post advertisements and positive comments about websites along with an URL. We perform detailed analysis of tasks trying to start information cascades on microblogging sites, and study the effectiveness of cascades as a function of the microblog social graph.

Third, we want to evaluate the end-to-end effectiveness of crowdturfing campaigns. To do so, we created accounts on one of our target systems, and initiated a number of benign campaigns that provide unsolicited advertisements for legitimate businesses. By bouncing clicks through our redirection server, we log responses to advertisements generated by our campaigns, allowing us to quantify their effectiveness. Our data shows that crowdturfing campaigns can be cost-effective at soliciting real user responses. Finally, we study and compare the source of workers on crowdturfing sites in different countries. We find that crowdturfing workers easily cross national borders, and workers in less-developed countries often get paid through global payment services for performing tasks affecting US-based networks. This suggests that the continuing growth of crowdturfing systems poses a real threat to U.S.-based online communities such as Facebook, Twitter, and Google+.

This study is the one of the first to examine the organization and effectiveness of large-scale crowdturfing systems on the Internet. These systems have already established roots in other countries, and are responsible for producing fake social network accounts that look indistinguishable from those of real users [244]. A recent study shows that similar types of behavior are also on the rise in the US-based Freelancer site [157]. Understanding the operation of these systems from both financial and technical angles is the first step to developing effective defenses to protect today's online social networks and online communities.

3.1.2 Crowdturfing Overview

In this section, we introduce the core concepts related to crowdturfing. We start by defining crowdturfing and the key players in a crowdturfing campaign. Next, we present two different types of systems that are used to effect crowdturfing campaigns on the Internet: *distributed* and *centralized*. Measurements of a distributed crowdturfing system show that it is significantly less popular with users than centralized systems. Thus we focus on understanding centralized crowdturfing systems in the remainder of our study.

3.1.2.1 Introduction to Crowdturfing

The term *crowdturfing* is a portmanteau of "crowd-sourcing" and "astroturfing." Astroturfing refers to information dissemination campaigns that are sponsored by an organization, but are obfuscated so as to appear like spontaneous, decentralized "grass-roots" movements. Astroturfing campaigns often involve spreading legally grey, or even illegal, content, such as defamatory rumors, false advertising, or suspect political messages. Although astroturfing pre-

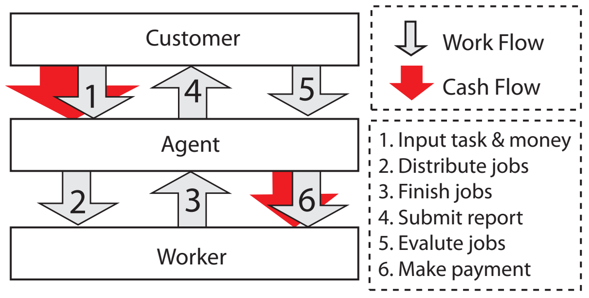


Figure 3.1: Work and cash flow of a crowdturfing campaign.

dates the Internet, the ability to quickly mobilize large groups via crowd-sourcing systems has drastically increased the power of astroturfing. We refer to this combined threat as *crowdturfing*. Because of its use of real human users, crowdturfing poses an immediate threat to existing security measures that protect online communities by targeting automated scripts and bots.

Crowdturfing campaigns on the Internet involve three key actors:

1. *Customers:* Individuals or companies who initiate a crowdturfing campaign. The customer is responsible for paying for the monetary costs, and are typically are either related to or themselves the beneficiaries of the campaign.
2. *Agents:* Intermediaries who take charge of campaign planning and management. The agent is responsible for finding, managing, and distributing funds to workers to accomplish the goals of the campaign.
3. *Workers:* Internet users who answer calls by agents to perform specific tasks in exchange for a fee.

Each campaign is structured as a collection of *tasks*. For example, a campaign might entail generating positive sentiment for a new restaurant. In this case, each task would be "post a single (fake) positive restaurant review online." Workers who complete tasks generate *submissions* that include evidence of their work. The customer/agent can then verify that the work was done to their satisfaction. In the case of the restaurant review campaign, submissions are screenshots of or URLs pointing to the fake reviews. Ideally, there is a one-to-one mapping between tasks and submissions. However, not all tasks may be completed, and submissions may be rejected due to lack of quality. In these cases, the number of submissions will not match the number of tasks for a given campaign.

The process for a crowdturfing campaign is shown in Figure 3.1. Initially, a customer brings the campaign to an agent and pays them to carry it out (1). The agent distributes individual tasks among a pool of workers (2), who complete the tasks and return submissions back to the agent (3). The agent passes the submissions back to the customer (4), who evaluates the work. If the customer is satisfied they inform the agent (5), who then pays the workers (6).

3.1.2.2 Crowdturfing Systems

Crowdturfing systems are instances of infrastructure used to connect customers, agents, and workers to enable crowdturfing campaigns. These systems are generally created and maintained by agents, and help to streamline the process of organizing workers, verifying their work, and distributing payments.

We have observed two different types of crowdturfing systems in the wild: distributed and centralized. We now describe the differences between these two structures, highlighting their respective strengths and weaknesses. Crowdturfing systems are similar to crowd-sourcing systems like Amazon's Mechanical Turk, with the exception that they accept tasks that are unethical or illegal, and that they can utilize distributed infrastructures.

Distributed Architecture. Distributed crowdturfing systems are organized around small instant message (IM) groups, mailing lists, or chat rooms hosted by group *leaders*. As illustrated in Figure 3.2a, leaders act as middlemen between agents and workers, and organizes the workers.

The advantage of distributed crowdturfing systems is that they are resistant to external threats, like law-enforcement. Individual forums and mailing lists are difficult to locate, and

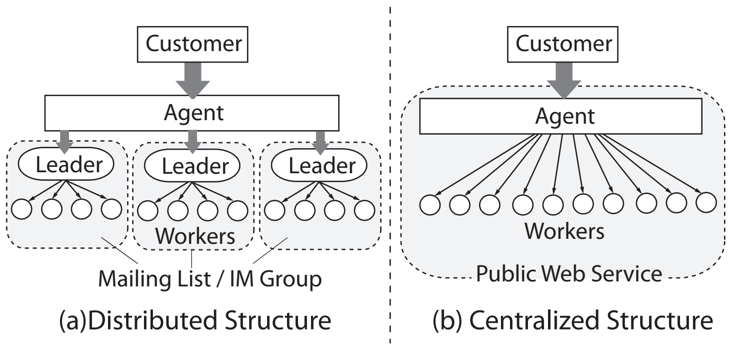


Figure 3.2: Two different crowdturfing system structures.

they can be dissolved and reconstituted elsewhere at any time. Furthermore, sensitive communications, such as payment transfers, occur via private channels directly between leaders and workers, and thus cannot be observed by third parties.

However, there are two disadvantages to distributed systems that limit their popularity. The first is lack of accountability. Distributed systems do not have robust reputation metrics, leaving customers with little assurance that work will be performed satisfactorily, and workers with no guarantees of getting paid. The second disadvantage stems from the fragmented nature of distributed systems. Prospective workers must locate groups before they can accept jobs, which acts as a barrier-of-entry for many users. To test this, we located 14 crowdturfing groups in China hosted on the popular Tencent QQ instant messaging network. Despite the fact that these groups were well advertised on popular forums, they only hosted ≈2K total users. Over the course of several days of observation, each group only generated 28 messages per day on average, most of which was idle chatter. The conclusion we can draw from these measurements is that distributed crowdturfing systems are not very successful at attracting workers. As we will show in Section 3.1.3, centralized systems attract orders of magnitude more campaigns and workers.

Centralized Architecture. Centralized crowdturfing systems, illustrated in Figure 3.2b, are instantiated as websites that directly connect customers and workers. Much like Amazon's Mechanical Turk, customers post campaigns and offer rewards, while workers sign up to complete tasks and collect payments. Both customer and workers register bank information associated with their accounts, and all transactions are processed through the website. Centralized crowdturfing websites use reputation and punishment systems to incentivize customer and workers to behave properly. The primary role of the agent in centralized architectures is simply to maintain the website, although they may also perform verification of submissions at the behest of customers.

The advantage of centralized crowdturfing systems is their simplicity. There are a small number of these large, public websites, making them trivial to locate by customers and workers. Centralized software automates campaign management, payment distribution, and maintains per-worker reputation scores. These features streamline centralized crowdturfing systems, and reduce uncertainties for all involved parties.

The disadvantage of centralized crowdturfing systems is their susceptibility to scrutiny by third parties. Since these public sites allow anyone to sign up, they are easy targets for infiltration, which may be problematic for crowdturfing sites that operate in legally grey-areas. On the other hand, this disadvantage made it possible for us to crawl and analyze several large crowdturfing websites.

3.1.3 Campaigns, Tasks, and Revenue

We begin our analysis of crowdturfing systems, by analyzing the volume of campaigns, tasks, users, and total revenue processed by the largest known systems. We first describe the representative systems in our study along with our data gathering methodology. We then present detailed results addressing these questions.

3.1.3.1 Data Collection and General Statistics

While a number of crowdsurfing systems operate across the global Internet, the two largest and most representative systems are hosted on Chinese networks. Their popularity is explained by the fact that China has both the world's largest Internet population (485M) [165] and a moderately low per-capita income (≈$3,200/year) [131]. Crowdturfing sites in China connect dodgy PR firms to a large online user population willing to act as crowd-sourced labor, and have been used to spread false rumors and advertising [64, 130, 72]. This "Shui Jun" (water army), as it is commonly known, has emerged as a force on the Chinese Internet that authorities are only beginning to grapple with [79, 8].

This confluence of factors makes China an ideal place to study crowdturfing. In this section, we measure and characterize the two largest crowdturfing websites in China: Zhubajie (ZBJ, `zhubajie.com`) and Sandaha (SDH, `sandaha.com`). All data on these sites are public, and we were able to gather all data on their current and past tasks via periodic crawls of their campaign histories.

Zhubajie and Sandaha. The first site we crawled is Zhubajie (ZBJ), which is the largest crowd-sourcing website in China. As shown in Table 3.1, ZBJ has been active for five years, and is well established in the Chinese market. Customers post many different legitimate types of jobs to ZBJ, including requests for freelance design and programming, as well as Mechanical Turk-style "human intelligence tasks." However, there is a subsection of ZBJ called "Internet Marketing" that is dedicated solely to crowdturfing. ZBJ also has an English-language version hosted in Texas (`witmart.com`), but its crowdturfing subsection only has 3 campaigns to date. Unlike ZBJ, Sandaha (SDH) only provides crowdturfing services, and is four years younger than ZBJ.

Crawling Methodology. We crawled ZBJ and SDH in September, 2011 to gather data for this study. We crawled SDH in its entirety, but only crawled the crowdturfing section of

Website	Active Since	Total Campaigns (%)	Total Workers	Total Tasks	Total Submissions (%)	Total Accepted (%)	Total Money	Money for Workers	Money for Website (%)
Zhubajie (ZBJ)	Nov. 2006	76K (92%)	169K	17.4M	6.3M (36%)	3.5M (56%)	$3.0M	$2.4M	$595K (20%)
Sandaha (SDH)	March 2010	3K (88%)	11K	1.1M	1.4M (130%)	751K (55%)	$161K	$129K	$32K (20%)

Table 3.1: General information for two large crowdturfing websites.

ZBJ. Both sites are structured similarly, starting with a main page that links to a paginated list of campaigns, ordered reverse chronologically. Each campaign has its own page that gives pertinent information, along with links to another paginated list of completed submissions from workers. All information on both sites is publicly available, and neither site employs security measures to prevent crawling.

Our crawler recorded details of all campaigns and submissions on ZBJ and SDH. Campaigns are characterized by a description, start and end times, total number of tasks, total money available to pay workers, whether the campaign is completed, and the number of accepted and rejected submissions. It also includes details for each submission entered by workers, including the worker username and UID, a submission timestamp, one or more screenshots and/or URLs pointing to content generated by the worker, and a flag marking the submission as either accepted or rejected after review.

Both ZBJ and SDH make the complete history of campaigns available on their sites, which enables the crawler to collect data dating back to each site's inception. Table 3.1 lists the total number of campaigns on each site, as well as the percentage that were usable for our study. Data on some campaigns is incomplete because the customer deleted them or made them private. Other data could be missing because either the campaign only provided partial information (*e.g.* no task count or price per task), or the campaign was still ongoing at the time of our crawl. Incomplete campaigns only account for 8% of the total on ZBJ and 12% on SDH, and thus have little impact on our overall results. For clarity, we convert all currency values on ZBJ and SDH (Chinese Yuan) to US Dollars using an exchange rate of 0.1543 to 1.

General Statistics. Table 3.1 shows the high-level results from our crawls. ZBJ is older

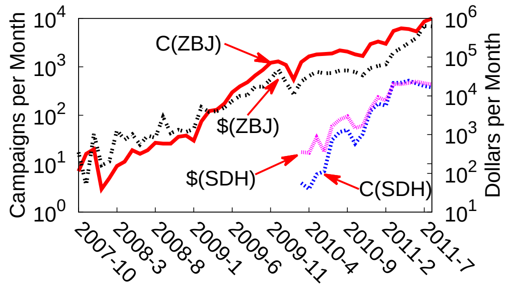


Figure 3.3: Campaigns, dollars per month.

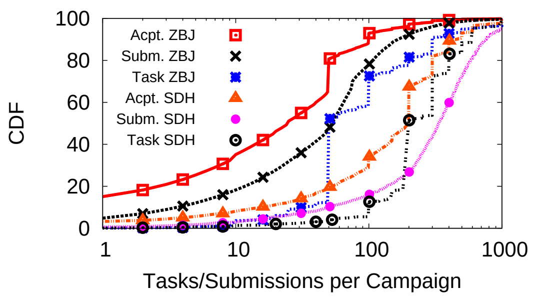


Figure 3.4: Tasks, submissions per camp.

and more well-established than SDH, hence it has attracted more campaigns, workers, and money. Campaigns on both sites each include many individual tasks, and task count is almost three orders of magnitude greater than number of campaigns. The number of submissions generated by workers in response to tasks is highly variable: on ZBJ only 36% of tasks receive submissions, whereas on SDH 130% of tasks receive submissions (*i.e.* there is competition among workers to complete the same tasks). Roughly 50% of all submissions are accepted.

Most importantly, more than \$4 million dollars have been spent on crowdturfing on ZBJ and SDH in the past five years. Both sites take a 20% cut of campaign dollars as a fee, resulting in significant profits for ZBJ, due to its high volume of campaigns. Furthermore, Figure 3.3 shows that the number of campaigns and total money spent are growing exponentially. The younger SDH has a growth trend that mirrors ZBJ, suggesting that it will reach similar levels of profitability within the next year. These trends indicate the rising popularity of crowdsurfing systems, and foreshadow the potential impact these systems will have in the very near future.

3.1.3.2 Campaigns, Tasks, and Workers

Figure 3.4 illustrates the high level breakdown of tasks and submissions on ZBJ and SDH. There are three lines corresponding to each site: tasks per campaign, submissions per campaign, and accepted submissions per campaign. Campaigns on ZBJ tend to have an order of magnitude fewer tasks than those on SDH. Although both sites only accept ≈50% of submis-

	Campaign Type	Num of Campaigns	\$/Camp.	\$/Task	Monthly Growth
ZBJ	Account Reg.	29,413 (39%)	\$71	\$0.35	16%
	Forum Post	17,753 (23%)	\$16	\$0.27	19%
	QQ Group	12, 969 (17%)	\$15	\$0.70	17%
	Microblog	4,061 (5%)	\$12	\$0.18	47%
	Blog Post	3,067 (4%)	\$12	\$0.23	20%
SDH	Forum Post	1,928 (57%)	\$48	\$0.19	40%
	QQ Group	473 (14%)	\$48	\$0.13	31%
	Q&A	463 (14%)	\$47	\$0.21	30%
	Blog Post	113 (3%)	\$49	\$0.19	21%
	Microblog	93 (3%)	\$49	\$0.27	42%

Table 3.2: The top five campaign types on ZBJ and SDH.

sions, the overabundance of submissions on SDH means that the number of accepted submissions closely tracks the required number of tasks, especially for campaigns with $>$100 tasks.

Campaign Types. Crowdturfing campaigns on ZBJ and SDH can be divided into several categories, with the five most popular listed in Table 3.2. These five campaign types account for 88% of all campaigns on ZBJ, and 91% on SDH.

"Account registration" refers to the creation of user accounts on a target website. Unlike what has been observed by prior work [157], these accounts are almost never used to automate the process of spamming. Instead, customers request this service to bolster the popularity of fledgling websites and online games, in order to make them appear well trafficked.

Four campaign types refer to spamming in specific contexts: QQ instant-message groups, forums, blogs, and microblogs (*e.g.* Twitter). Customers in China prefer to pay workers directly to generate content on popular websites, rather than purchasing accounts from workers and spamming through them. Note, that QQ and forums represent a larger percentage of campaigns because their existence predates microblogs, which have only become popular in China in the last year [165]. The last column of Table 3.2 shows the average monthly growth in number of campaigns, and shows that microblogs campaigns are growing faster than all other

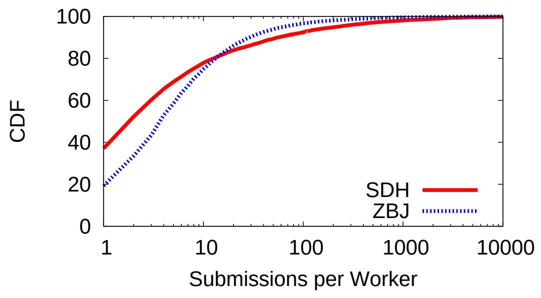


Figure 3.5: Submissions per worker.

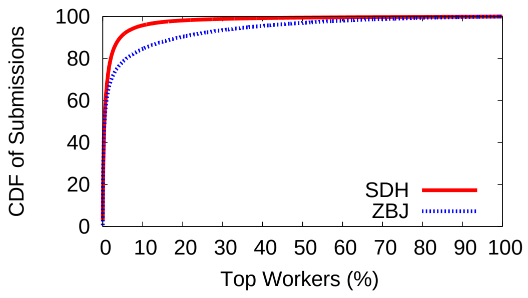


Figure 3.6: Submissions by top workers.

top-5 categories in both ZBJ and SDH. As the popularity of social networks and microblogs continues to grow, we expect to see more campaigns targeting them.

Finally, “Q&A” involves posting and answering questions on social Q&A sites like Quora (`quora.com`). Workers are expected to answer product-related questions in a biased manner, and in some cases post dummy questions that are immediately answered by other colluding workers.

Worker Characteristics. We now focus our discussion on the behavior of workers on crowdturfing websites. Figure 3.5 shows that the total number of submissions per worker (including both accepted and rejected submissions) varies across the worker population, and even between ZBJ and SDH. Roughly 40% of SDH workers only complete a single task, compared to 20% on ZBJ. The average worker on both sites complete around 5-7 tasks each.

Figure 3.5 also reveals that a small percentage of extremely prolific workers (especially on SDH) generate hundreds, even thousands, of submissions. Figure 3.6 plots the percentage of submissions from top workers ordered from most to least prolific. The distribution is highly skewed in favor of these career crowdturfers, who are responsible for generating ≈75% of submissions.

We now examine the temporal aspects of worker behavior. Figure 3.7 plots the time difference between a campaign getting posted online, and the first submission from a worker. On SDH, 50% of campaigns become active within 24 hours, whereas on ZBJ (with its larger

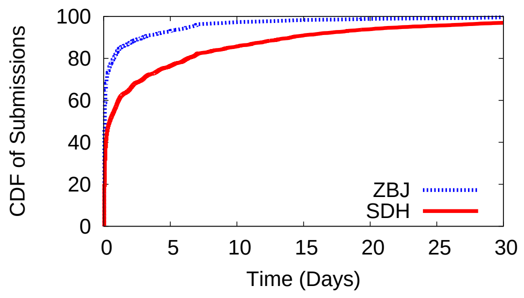


Figure 3.7: Time to first response.

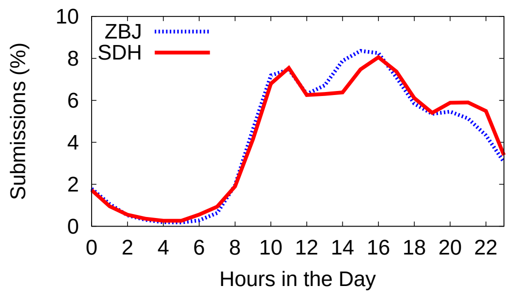


Figure 3.8: Daily submissions.

worker population) 75% of campaigns become active within 24 hours. However, some campaigns take significantly longer to ramp up: up to 15 days on ZBJ, and 30 days on SDH. As we discuss in Section 3.1.3.3, these slow moving campaigns have very specific requirements that cannot be met by the vast majority of workers.

Figure 3.8 shows the correlation between time of day and number of submissions on ZBJ and SDH. Most submissions happen during the workday and in the evening. Slight drops around lunch and dinner are also visible. This pattern confirms that submissions are generated by human beings, and not automated bots.

3.1.3.3 Money

We now explore the monetary reward component of crowdturfing systems. As is common on crowd-sourcing systems like Mechanical Turk, workers on ZBJ and SDH make a tiny fee for each accepted submission. As shown in Figure 3.9, the vast majority of workers on ZBJ and SDH earn \$0.11 per submission, although ≈20% of submissions command higher prices than this. Workers must complete many submissions in order to earn substantial pay, leading to the prolific submission habits of career crowdturfers seen in Figure 3.6. Note that this is a very different model from bid-for-tasks systems like the recent Freelancer study [157].

The total amount of money earned by most workers on ZBJ and SDH is very small. As illustrated in Figure 3.10, close to 70% of workers earn less than \$1 for their efforts. The re-

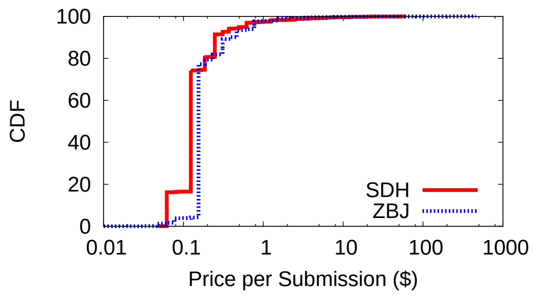


Figure 3.9: Submission prices.

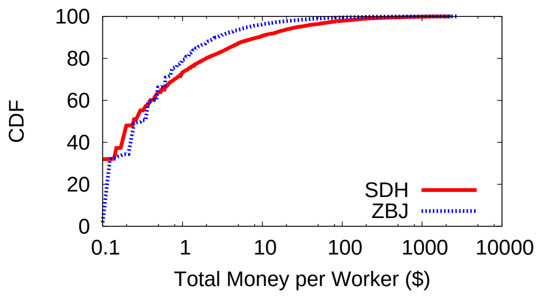


Figure 3.10: Money per worker.

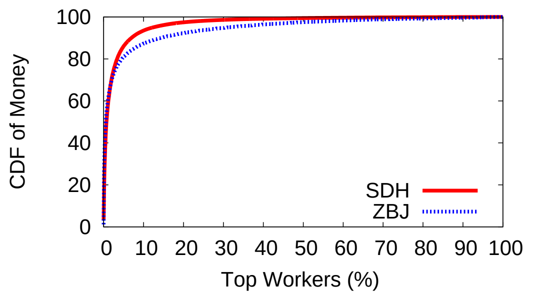


Figure 3.11: Money earned by top workers.

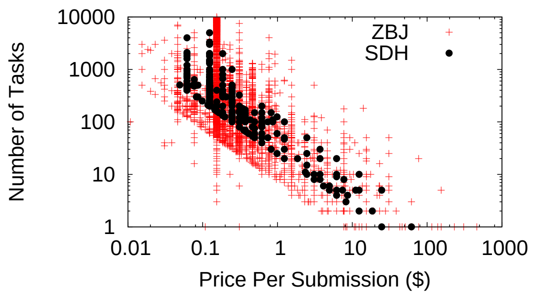


Figure 3.12: # of tasks vs. submission price.

maining 30% of workers earn between $1 and $100, making crowdturfing a potentially rewarding part-time job to supplement their core income. For a very small group of workers (0.4%), crowdturfing is a full-time job, earning rewards in the $1,000 dollar range. Not surprisingly, the distribution of monetary rewards matches this distribution. As seen in Figure 3.11, the top 5% of workers take home 80% of the proceeds on ZBJ and SDH. Clearly, a hard-core contingent of career crowdturfers is taking the bulk of the reward money by quickly completing many submissions.

Task Pricing. The goal and budget of each crowdturfing campaign affects the number and price of tasks in that campaign. Figure 3.12 plots the correlation between the number of tasks in a campaign, versus the price per submission the customer is willing to pay. The vast majority of campaigns with 1K-10K tasks call for generating numerous "tweets" on microblog sites. We examine these tasks in more detail in Section 3.1.4.

Although the vast majority of campaigns call for many tasks with low price per submission, Figure 3.12 reveals that there is a small minority of well paying tasks. In many cases, these campaigns only include a single task that can earn an accepted submission $\geq$\$100 dollars. We examined the 158 outlying tasks that earned $\geq$\$10 and determined that they include a large range of very strange campaigns, some prominent examples include:

- *Pyramid Schemes:* Workers recruit their friends into a pyramid scheme to receive a large payment.
- *Commissioned Sales:* Workers sell products in order to receive a percentage of the sales.
- *Dating Sites:* Workers crawl OSNs and clone the profiles of attractive men and women onto a dating site.
- *Power-Users:* These tasks call for a single worker who owns a powerful social network account, well-read blog, or works for a news service to generate a story endorsing the customer.

3.1.4 Crowdturfing on Microblogs

In this section, we study the broader impact of crowdturfing by measuring the spread of crowdturf content on microblogging sites. We gather data from Sina Weibo, the most popular microblogging social network in China that has the same look and feel as Twitter. We study Weibo for two reasons. First, as shown in Table 3.2, microblogging sites and Weibo in particular are very popular targets for crowdturfing campaigns. Second, the vast majority of information on Weibo (*i.e.* “tweets” and user profile information) is public, making it an ideal target for measurement and analysis.

We begin by introducing Weibo and our data collection methodology. Next, we examine properties of crowdturfing tasks and workers on Weibo. Finally, we gauge the success of campaigns across the social network by analyzing the spread of crowdturfing content.

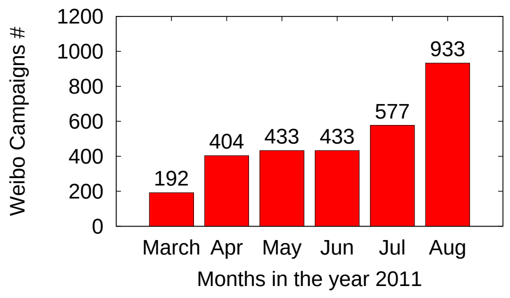


Figure 3.13: Weibo campaigns in 2011.

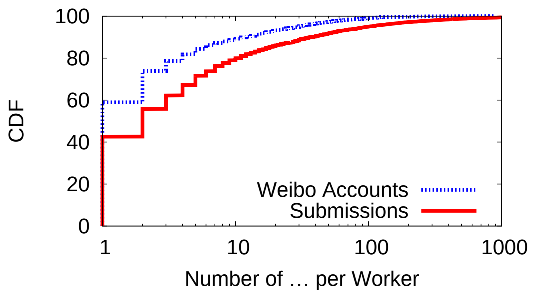


Figure 3.14: Submissions and accounts.

3.1.4.1 Weibo Background and Data Collection

Founded in August 2009, Sina Weibo is the most popular microblogging social network in China, with more than 250 million users as of October 2011 [7]. Weibo has functionality identical to Twitter: users generate 140 character "tweets," which can be replied to and "retweeted" by other users. Users may also create directed relationships with other users by *following* them.

We focus our study of Weibo campaigns from ZBJ, because ZBJ has the most microblogging campaigns by far. Of the 4,061 microblogging campaigns on ZBJ, 3,145 target Weibo. As shown in Figure 3.13, the number of Weibo campaigns on ZBJ mirrors Weibo's rapid growth in popularity in 2011.

The goal of crowdturfing campaigns on Weibo is to increase the customer's reach, and to spread their sponsored message throughout the social network. These goals lead to three task types: "pay per tweet," "pay per retweet," and "purchasing followers." The most common task type is retweeting, in which the customer posts a tweet and then pays workers to retweet it. Alternatively, customers may pay workers to generate their own tweets, laden with specific keywords and URLs, or to have their accounts follow the customer's for future messages.

To increase the power of their campaigns, customers prefer workers who use realistic, well-maintained Weibo accounts to complete tasks. Customers may not accept submissions from poor quality, *e.g.* easily detected or banned, Sybil accounts. Conversely, workers who control popular accounts with many followers can earn more per task than worker accounts with average popularity.

Data Collection. Understanding the spread of crowdturfing content on Weibo requires identifying *information cascades* [128]. Each cascade is characterized by an *origin post* that initiates the cascade, and retweets that further propagate the information. Cascades form a directed tree with the origin post at the root. In *crowdturfing cascades*, the origin post is always generated by a customer or a worker, but retweets can be attributed to workers and normal Weibo users. Each campaign is a forest of cascade trees.

We crawled Weibo in early September, 2011 to gather data on the spread of crowdturf content. The crawler was initially seeded with URLs that matched campaigns already found on ZBJ, and used simple content analysis to determine if each worker submission was an origin post or a retweet in order to differentiate between "pay to tweet" and "pay to retweet" tasks. In the latter case, the crawler fetched the origin post using information embedded in the retweet.

Our crawler targets the mobile version of the Weibo site because it lists all retweets of a given origin post on a single page, including the full path of multi-hop retweets. The crawler recorded the total number of tweets, followers, and users followed by each user involved in crowdturfing cascades. Unfortunately, Weibo only divulges the first 1K followers for each user, so we are unable to fully reconstruct the social graph.

Overall, our crawler collected 2,869 campaigns involving 1,280 customers. These campaigns received submissions from more than 12,000 Weibo accounts, and reached more than 463,000 non-worker users. Among these, 2% of worker accounts were inaccessible, and were presumably banned by Weibo for spamming. 0.08% of the non-worker user accounts were inaccessible, and all customer accounts remained active. "Pay per tweet" campaigns initiated 25,000 cascades, while "pay per retweet" campaigns triggered 5,000 cascades. We ignore "purchase followers" campaigns, since they do not generate crowdturfing cascades.

To get a baseline understanding of normal Weibo user accounts, we performed a snowball crawl of Weibo's social graph in October 2011. The result is profile data for 6 million "normal"

Weibo users.

3.1.4.2 Weibo Account Analysis

We begin by examining and comparing the characteristics of Weibo accounts controlled by workers and customers to those of normal Weibo users. As shown in Figure 3.14, the number of accounts controlled by each worker follows the same trend as submissions per worker. This is intuitive: workers need multiple accounts in order to make multiple submissions to a single campaign. Hence, professional crowdturfers who generate many submissions need to control a commensurate number of accounts. In absolute terms, we observe 14,151 accounts controlled by 5,364 ZBJ workers. The top 1% of workers each control $\geq$100 accounts, but the average worker controls only $\approx$6 accounts.

Comparison to Normal Accounts. We now compare characteristics of worker's and customer's accounts to normal users. We find that each account type tweets with the same frequency. This suggests that workers and customers are both careful not to overwhelm their followers with spam tweets.

Previous work on Sybil detection on OSNs showed that *follow rate* is an effective metric for locating aberrant accounts [206]. A user's follow rate is defined as the ratio of followers to users followed. Sybils often attempt to gain followers by following many other users and hoping they reciprocate. Thus Sybils have follow rates $<$1, *e.g.* they follow more users than they have followers.

Figure 3.15 shows the follow rates for different Weibo account types. Surprisingly, normal users have the lowest follow rates. Most worker accounts have follow rates $\approx$1, allowing them to easily blend in. This may represent a conscious effort on the part of workers to make their Weibo accounts appear "normal" so that they will evade automatic Sybil detectors. Customers tend to have follow rates $>$1. This makes sense, since customers tend to be commercial entities,

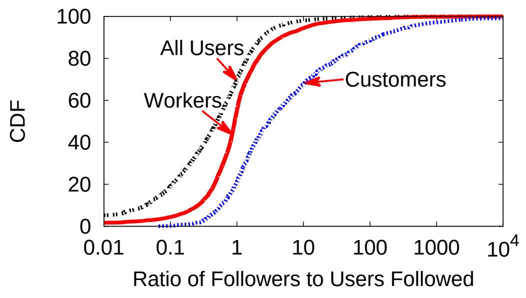


Figure 3.15: Follow rates for Weibo users.

and are thus net information disseminators rather than information consumers.

3.1.4.3 Information Dissemination on Weibo

Much work has studied how to optimize information dissemination on social networks. We analyze our data to evaluate the level of success in crowdturfing cascades, and whether there are factors that can predict the success of social crowdturfing campaigns.

Campaign Analysis. We start by examining the number of *messages* generated by crowdturfing campaigns on Weibo. We define a message as a single entry in a Weibo timeline. A tweet from a single user generates f messages, where f is their number of followers. The number of messages in a campaign is equal to the number of messages generated by the customer, workers, and any normal users who retweet the content. Total messages per campaign represents an upper bound on the *audience size* of that campaign. Since we have an incomplete view of the Weibo social graph, we cannot quantify the number of duplicate messages per user.

Figure 3.16 shows the CDF of messages generated by Weibo campaigns. 50% of campaigns generate $\leq$146K messages, and 8% manage to breach the 1M-message milestone. As expected, workers are responsible for the vast majority of messages, *i.e.* there are very few retweets. Considering the low cost of these campaigns, however, these raw numbers are nonetheless impressive.

Next, we want to examine the depth of crowdturfing cascades. Figure 3.17 plots the depth

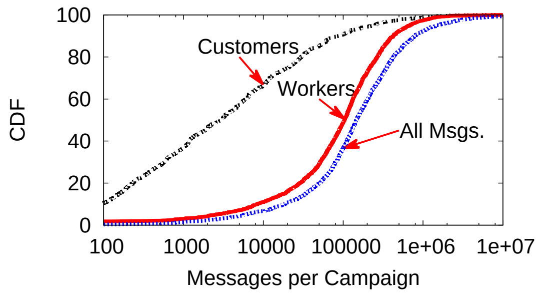


Figure 3.16: Messages per campaign.

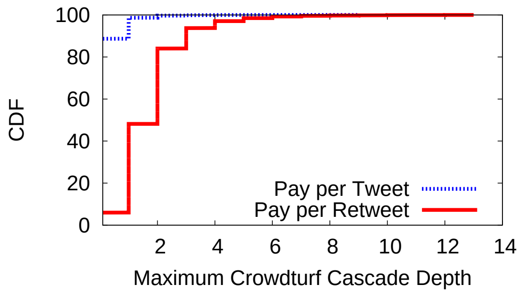


Figure 3.17: Crowdturfing cascades.

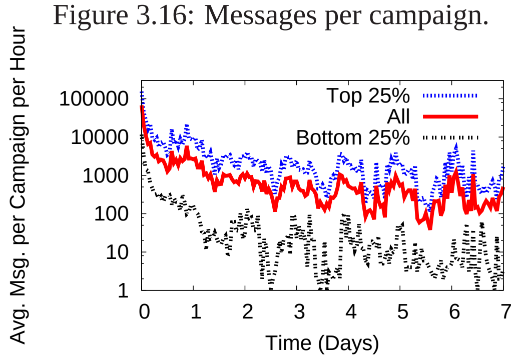


Figure 3.18: Message creation over time.

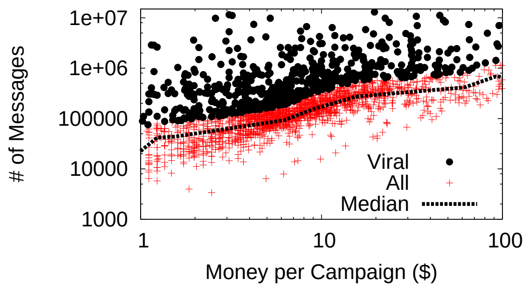


Figure 3.19: Cost of Weibo campaigns.

of cascades measured as the height of each information cascade tree. Pay-per-tweet campaigns are very shallow, *i.e.* worker's tweets are rarely retweeted by normal users. In contrast, pay-per-retweet campaigns are more successful at engaging normal users: 50% reach depths >2, *i.e.* they include at least one retweet from a normal user. One possible explanation for the success of pay per retweet is that normal users may place greater trust in information that is retweeted from a popular customer, rather than content authored by random worker accounts.

Next, we examine the temporal dynamics of crowdturfing campaigns. Figure 3.18 shows the number of messages generated per hour after each campaign is initiated. The "all" line is averaged across all campaigns, while the top- and bottom-25% lines focus on the largest and smallest campaigns (in terms of total messages). Most messages are generated during a campaigns' first hour (10K on average), which is bolstered by the high-degree of customers (who tend to be super-nodes), and the quick responses of career crowdturfers (see Figure 3.7). However, by the end of the first day, the message rate drops to ≈1K per hour. There is a two order of magnitude difference between the effectiveness of the top- and bottom-25% campaigns, although they both follow the same falloff trend after day 1.

Factors Impacting Campaign Success. We now take a look at factors that may affect the performance of crowdturfing cascades. The high-level question we wish to answer is: are there specific ways to improve the probability that a campaign goes *viral*?

The first factor we examine is the cost of the campaign. Figure 3.19 illustrates the number of messages generated by Weibo campaigns versus their cost. The median line, around which the bulk of campaigns are clustered, reveals a linear relationship between money and messages. This result is intuitive: more money buys more workers, who in turn generate more messages. However, Figure 3.19 also reveals the presence of *viral* campaigns, which we define as campaigns that generate at least two times more messages than their cost would predict. There are 723 viral campaigns scattered randomly throughout the upper portion of Figure 3.19. This shows that viral popularity is independent of campaign budget.

We look at whether specific workers are better at generating viral campaigns. We found that individual workers are not responsible for the success of viral campaigns. The only workers consistently involved in viral campaigns are career crowdturfers, who tend to be involved in *all* campaigns, viral or not.

Surprisingly, a small number of customers exhibit a consistent ability to start viral campaigns. Figure 3.20 plots the total number of campaigns started by each customer vs. the number that went viral, for all customers who started at least 1 viral campaign. The vast majority of customers initiate $\leq$3 campaigns, which makes it difficult to claim correlation when one or more go viral. However, the 20 customers (1.5%) in the highlighted region do initiate a significant number of campaigns, and they go viral $\geq$50% of the time. Since many of these customers do not actively participate in their own campaigns, this suggests that campaigns go viral because their content is of interest to Weibo users, perhaps because they are related to customers such as well-known actors or performers.

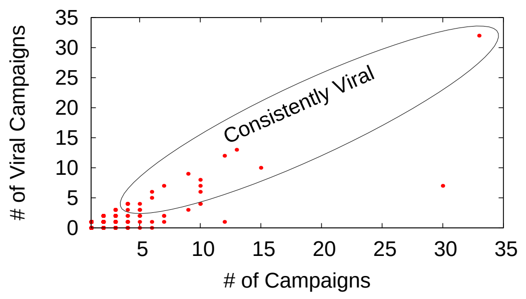


Figure 3.20: Viral campaigns per customer.

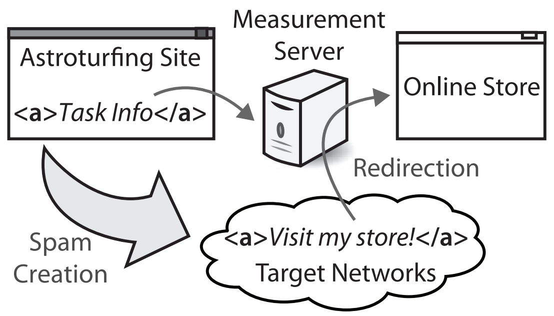


Figure 3.21: Crowdturfing data collection.

3.1.5 Active Experiments

Our next step to understanding crowdsurfing systems involves a look from the perspective of a paying customer on ZBJ. We initiate a number of benign advertising campaigns on different platforms and subjects. By redirecting the click traffic through a *measurement server* under our control, we are able to analyze the clicks of workers and of users receiving crowdturf content in real-time. We begin by describing our experimental setup before moving on to our findings, and conclude with a discussion of practical lessons we learned during this process.

3.1.5.1 Experimental Setup

Methodology. Figure 3.21 depicts the procedure we use to collect real-time data on crowdturfing clicks. The process begins when we post a new campaign to ZBJ that contains a brief description of the tasks, along with a URL (“Task Info” in Figure 3.21) that workers can click on to find details and to perform the tasks. The task details page is hosted on our measurement server, and thus any worker who wants to accept our tasks must first visit our server, where we collect their information (*i.e.* IP, timestamp, etc). Referring workers to task details on external sites is a common practice on ZBJ, and does not raise suspicion among workers.

Workers that accept our tasks are directed to post spam messages that advertise real online stores to one of three target networks: Weibo, QQ instant message groups, and discussion forums. The posted messages urge normal users to click embedded links (“Visit my store!” in

Campaign	Network	Subm.	Time	Msgs.	Clicks	W/IP
iPhone4S	Weibo	47	45min	197K	204	24/54
	QQ	41	6hr	35K	244	34/36
	Forums	71	3day	N/A	43	40/22
Maldives	Weibo	108	3h	220K	28	35/30
	QQ	118	4h	46K	187	24/29
	Forums	123	4h	N/A	3	18/11
Raffle	Weibo	131	2h	311K	47	67/38
	QQ	131	6day	60K	78	29/33
	Forums	124	1day	N/A	0	28/9
OceanPark	Weibo	204	4day	369K	63	204/99

Table 3.3: Results from our crowdturfing campaigns.

Figure 3.21) that take them to our measurement server. The measurement server records some user data before transparently redirecting them to the real online store.

We took care to preserve the integrity of our experimental setup. Because some Chinese Internet users have limited access to websites hosted outside of mainland China, we placed our measurement server in China, and only advertised legitimate Chinese e-commerce sites. In addition, we also identified many search engines and bots generating clicks on our links, and filtered them out before analyzing our logs.

Campaign Details. In order to experiment with a variety of topics and venues, we posted nine total campaigns to ZBJ in October 2011. As shown in Table 3.3, we created three different advertising campaigns (*iPhone4S*, *Maldives*, and *Raffle*), and targeted each at three distinct networks. We discuss a fourth campaign, *OceanPark*, later in the section.

The first campaign promotes an unofficial iPhone dealer who imports iPhones from North America and sells them in China. We launched this campaign on October 4, 2011, immediately after Apple officially unveiled the iPhone 4S. In the task requirements, we required workers to post messages advertising a discount price from the dealer on the iPhone 4S ($970).

The second campaign tried to sell a tour package to the Maldives (a popular tourist destination in China). The spam advertises a 30% group-purchase discount offered by the seller that saves $600 on the total trip price ($1542 after discount). The third campaign tells users about an online raffle hosted by a car company. Anyone could participate in the raffle for free, and the prizes were 200 pre-paid calling cards worth $4.63 each.

All campaigns shared the same set of baseline requirements. Each campaign had a budget of $15 on each target network, and workers had a time limit of 7 days to perform tasks. The desired number of tasks was set to either 50 or 100, depending on the campaign type. Submissions were not accepted if the content generated by the worker was deleted by spam detection systems within 24 hours of creation. These baseline requirements closely match the expected norms for campaigns on ZBJ (see Figure 3.4 and Table 3.2).

We applied additional requirements for campaigns on specific networks. For campaigns on the QQ instant messaging network, workers were required to generate content in groups with a minimum of 300 members. For campaigns on user discussion forums, workers were only allowed to post content on a predefined list of forums that receive at least 1,000 hits per day.

Each campaign type had additional, variable requirements. For Maldives and Raffle campaigns, the price per task was set to $0.154, meaning 100 submissions would be accepted. However, the price for iPhone4S tasks was doubled to $0.308 with an expectation of 50 submissions. iPhone 4S tasks were more challenging for two reasons. On Weibo, workers were required to tweet using accounts with at least 3,000 followers. On QQ, workers needed to spam two groups instead of one. Finally, on forums, the list of acceptable sites was reduced to only include the most popular forums.

3.1.5.2 Results and Analysis

Table 3.3 lists the high level results of from our crowdturfing campaigns, including 9 short campaigns and the “OceanPark” campaign. Seven of the short campaigns received sufficient submissions, and six were completed within a few hours (Time column). Interestingly, workers continued submitting to campaigns even after they were “full,” in the hopes that earlier sub-

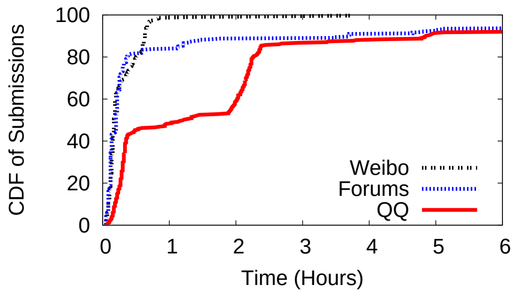


Figure 3.22: Response time of ZBJ workers.

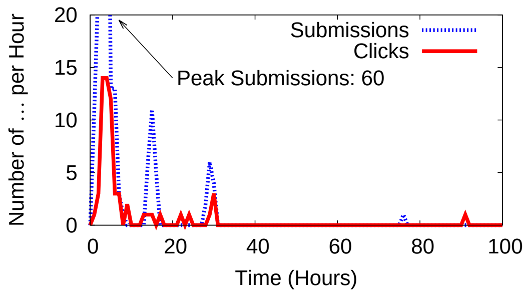


Figure 3.23: Long campaign characteristics.

missions would be rejected, and they would claim the reward. In total, the short campaigns garnered 894 submissions from 224 distinct workers.

Figure 3.22 shows the response times of workers for campaigns targeting different networks. We aggregate the data across campaign types rather than networks because workers' ability to complete tasks is based on the number of accounts they control on each network. More than 80% of submissions are generated within an hour for Weibo and forum campaigns, and within six hours for QQ.

The "Msgs" column lists the number of *messages* generated by each campaign. For Weibo campaigns, we calculate messages using the same methodology as in Section 3.1.4. For QQ campaigns, messages are calculated as the number of users in all QQ groups that received spam from our workers. We cannot estimate the number of messages for forums because we do not know how many users browse these sites.

We can understand the effectiveness of different crowdturfing strategies by comparing the number of messages generated to the number of clicks (responses by normal users, "Clicks" column in Table 3.3). We see that QQ campaigns are the most effective, and generate more clicks than Weibo campaigns despite generating only 1/5 as many messages as Weibo. One possible reason is that QQ messages pop-up directly on users' desktops, leading to more views and clicks. Tweets on Weibo, on the other hand, are not as invasive, and may get lost in the flood of tweets in each user's timeline. Forums perform the worst of the three, most likely because admins on popular forums are diligent about deleting spammy posts.

Finally, we try to detect the presence of Sybil accounts (multiple accounts controlled by one user) on crowdturfing sites. Column "W/IP" in Table 3.3 compares the number of distinct workers (W) to the number of distinct IPs (IP) that click on the "Task Info" link (see Figure 3.21) in each campaign. If $W > IP$, then not all ZBJ workers clicked the link to read the instructions. This suggests that multiple ZBJ worker accounts are controlled by a single user, who viewed the instructions once before completing tasks from multiple accounts. Our results show that $W > IP$ for 66% of our campaigns. Thus, not only do crowdturfers utilize multiple accounts on target websites to complete tasks (Figure 3.14), but they also have multiple accounts on crowdturfing sites themselves.

Long Campaigns. The campaigns we have analyzed thus far all required $\leq$100 tasks, and many were completed within about an hour by workers (see Figure 3.22). These short campaigns favor career crowdturfers, who control many accounts on target websites and move rapidly to generate submissions.

To observe the actions of less prolific workers, we experimented with a longer campaign that required 300 tasks. This campaign included an additional restriction to limit career crowdturfers: each ZBJ worker account could only submit once. The goal of the campaign was to advertise discount tickets to an ocean-themed amusement park in Hong Kong on Weibo. This campaign is listed as *OceanPark* in Table 3.3.

Figure 3.23 plots the number of worker submissions and clicks from Weibo users over time for the OceanPark campaign. Just as in previous experiments, the first 100 submissions were generated within the first few hours. Clicks from users on the advertised links closely track worker submission patterns. Overall, 191 submissions were received on day one, 11 more on day two, and 2 final submissions on day four, for a total of 204 submissions. This indicates that there are $\approx$200 active Weibo workers on ZBJ: if there were more, they would have submitted to claim one of the 97 incomplete tasks in our campaign.

Discussion. Our real-world experiments demonstrate the feasibility of crowd-sourced spamming. The iPhone4S and Maldives campaigns were able to generate 491 and 218 click-backs (respectively) while only costing $45 each. Considering that the iPhone 4S sells for $970 in China, and the Maldives tour package costs $1,542, just a single sale of either item would be more than enough to recoup the entire crowdturfing fee. The *cost per click* (CPC) of these campaigns are $0.21 and $0.09, respectively, which is more expensive than observed CPC rates ($0.01) for traditional display advertising on the web [204]. However, with improved targeting (*i.e.* omitting underperformers like forum spam) the costs could be reduced, bringing CPC more in line with display advertising.

Our Maldives campaign is a good indicator of the effectiveness of crowdturfing. The tour website listed 4 Maldives trips sold to 2 people in the month before our campaign. However, the day our Maldives campaign went live, 11 trips were sold to 2 people. In the month after our campaign, no additional trips were sold. While we cannot be sure, it is likely that the 218 clicks from our campaign were responsible for these sales.

3.1.6 Crowdturfing Goes Global

In previous sections, we focused on the crowdturfing market in China. We now take a global view and survey the market for crowdturfing systems in the U.S. and India. Additional crawls conducted by us, as well as prior work from other researchers, demonstrates that crowdturfing systems in the U.S. are very active, and are supported by an international workforce.

Mechanical Turk. Although prior work has found that 41% of tasks on Mechanical Turk were spam related in 2010 [107], our measurements indicate that this is no longer the case. We performed hourly crawls of Mechanical Turk for one month in October 2011, and used keyword analysis to classify tasks. As shown in Table 3.4, crowdturfing now only accounts for only 12% of campaigns.

Website	Cam-paigns.	% Crowd-turfing	Tasks	$ per Subm.
Amazon Turk (US)	41K	12%	2.9M	$0.092
ShortTask* (US)	30K	95%	527K	$0.096
MinuteWorkers (US)	710	70%	10K	$0.241
MyEasyTask (US)	166	83%	4K	$0.149
Microworkers (US)	267	89%	84K	$0.175
Paisalive (India)	107	N/A	N/A	$0.01

Table 3.4: Details of U.S. and Indian crowd-sourcing sites. Data encompasses one month of campaigns, except ShortTask which is one year.

Other U.S. Based Sites. However, the drop in crowdturfing on Mechanical Turk does not mean this problem has gone away. Instead, crowdturfing has just shifted to alternative websites. For example, recent work has shown that 31% of the jobs on Freelancer over the last seven years were related to search engine optimization (SEO), Sybil account creation, and spam [157]. Many SEO products are also available on eBay: trivial keyword searches turn up many sellers offering bulk Facebook likes/fans and Twitter followers.

To confirm this finding, we crawled four U.S. based crowd-sourcing sites that have been active since 2009. Since they do not provide information on past tasks, we crawled MinuteWorkers, MyEasyTask, and Microworkers once a day during the month of October 2011. ShortTask does provide historical data for tasks going back one year, hence we only crawled them once. As shown in Table 3.4, keyword classification reveals that between 70-95% of campaigns on these sites are crowdturfing. We manually verified that the remaining campaigns were not malicious. The types of campaigns on these sites closely matches the types found on Freelancer, *i.e.* the most prevalent campaign type is SEO [157].

Sites like ShortTask, Microworkers, and MyEasyTask fill two needs in the underground market. First, they do not enforce any restrictions against crowdturfing. This contrasts with Mechanical Turk, which actively enforces policies against spammy jobs [73]. Second, these sites enable a truly international workforce by supporting a wide range of payment methods.

Amazon requires workers to have U.S. bank accounts, or to accept cheques in Indian Rupees, and hence most “turkers” are located in the US (46.8%) and India (34%) [106]. However, alternative crowd-sourcing sites support payments through systems like Paypal and E-Gold, which makes them accessible to non-U.S. and non-Indian workers. For example, Microworkers come from Indonesia (18%), Bangladesh (17%), Philippines (5%), and Romania (5%) [101]. Freelancers are also located in the United Kingdom and Pakistan [157].

Paisalive. We located one crowdturfing site in India called Paisalive that takes globalization even further. As shown in Table 3.4, Paisalive is very small and the wages are very low compared to other services. However, the interesting feature of Paisalive is that it is e-mail based: workers sign up on the website, and afterwards all task requests and submissions are handled through e-mail. This design is geared towards enabling workers in rural populations constrained by low-bandwidth, intermittent Internet connectivity.

3.1.7 Summary of Results

In summary, we contribute to the growing pool of knowledge about malicious crowdsourcing systems. Our analysis of the two largest crowdturfing sites in China reveals that $4 million dollars have already been spent on these two sites alone. The number of campaigns and dollars spent on ZBJ and SDH are growing exponentially, meaning that the problems associated with crowdturfing will continue to get worse in the future.

We measure the real-world ramifications of crowdturfing by looking at spam dissemination on Weibo, and by becoming active customers of ZBJ. Our results reveal the presence of career crowdturfers that control thousands of accounts on OSNs, and manage them carefully by hand. We find that these workers are capable of generating large information cascades, while avoiding the security systems that are designed to catch automated spam. We also observe that this spam is highly effective, driving hundreds of clicks from normal users.

Finally, our survey of crowdturfing sites in the U.S. and elsewhere demonstrates the global nature of this problem. Unscrupulous crowd-sourcing sites, coupled with international payment systems, have enabled a burgeoning crowdturfing market that targets U.S. websites, fueled by a global workforce. As part of ongoing work, we are exploring the design and quantifying the effectiveness of both passive and active defenses against these systems.

3.2 Defense and Adversarial Attacks

3.2.1 Introduction

Thus far, Section 3.1 shows that crowdturfing is an emerging security threat to online users and online services. Next, we explore possible defense techniques against crowdturfing using machine learning techniques.

Today's online 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 [179, 182], network intrusion attacks [129, 250], and Internet worms [164]. More recently, researchers have used them to model and detect malicious users in online services, *e.g.* Sybils in social networks [206, 244], scammers in e-commerce sites [249] and fraudulent reviewers on online review sites [169].

Detection of Crowdturfing. In the rest of this chapter, we explore the possibility of using machine learning techniques to detect crowdturfing activities. 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 Attacks. 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 [51, 68, 160], 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.

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.

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.

3.2.2 Datasets and Methodology

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

3.2.2.1 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) [230]. Both sites leave records of campaigns publicly visible to recruit new workers, making it possible for us to crawl their data for analysis.

Crowdturfing on Weibo. Sina Weibo is China's most popular microblogging social network with over 500 million users [167]. 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

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 3.5: Dataset summary.

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 [21]. A recent statement (April 2014) from a Weibo administrator shows that Weibo has already begun to take action against crowdturfing spam [26].

3.2.2.2 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.

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[1]. We ran

[1]Snowball crawls start from an initial set of seed nodes, and runs breadth-first search to find all reachable nodes in the social graph [37].

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[2] 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. 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 3.5.

3.2.2.3 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 (§3.2.3).

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

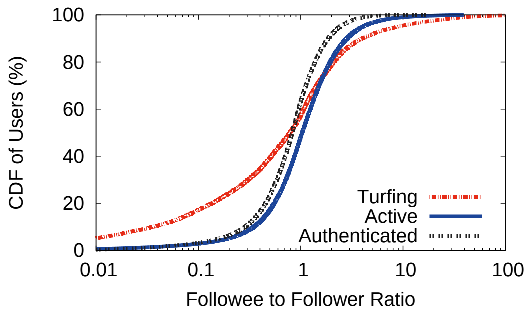


Figure 3.24: Followee-to-Follower ratio.

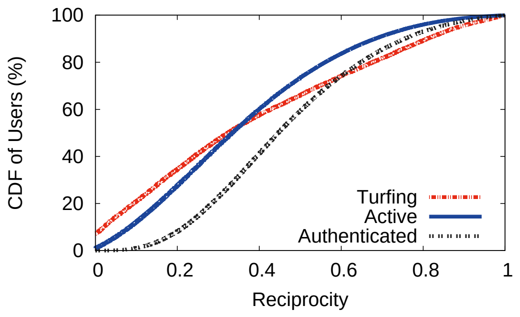


Figure 3.25: Reciprocity.

- 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) (§3.2.4).
- 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).

3.2.3 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 §3.2.4.

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

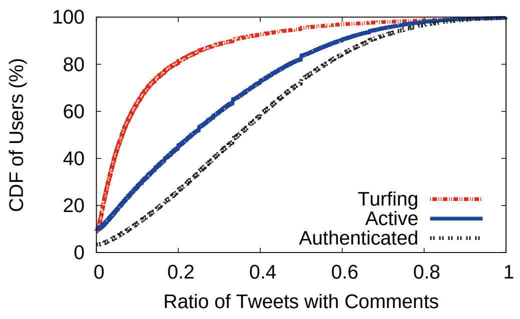


Figure 3.26: Ratio of commented tweets.

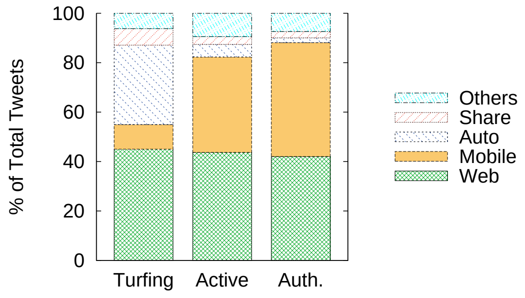


Figure 3.27: Tweet client usage.

to identify malicious spammers [46, 213, 241]. 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 3.24), and their reciprocity is very close to those of normal users (Figure 3.25). 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 [46, 213], 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 [225].

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 3.26 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

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 3.6: High-level categories for tweeting clients.

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 tools, and third-party websites via "share" buttons (Table 3.6). Figure 3.27 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 [230].

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

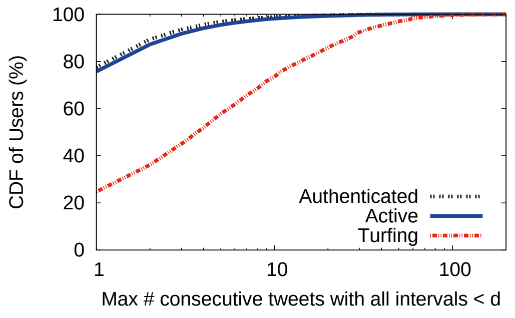


Figure 3.28: Max size of tweeting burst (threshold $d = 1$ minute).

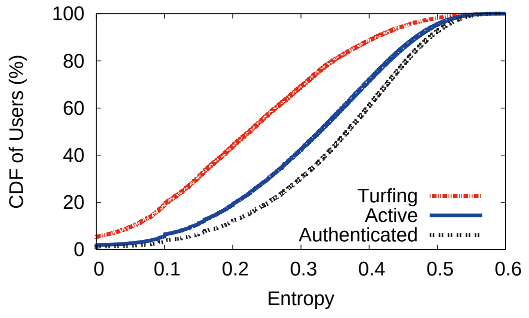


Figure 3.29: Normalized entropy of tweeting inter-arrival time.

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 3.28 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 [81], 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 [81]. Figure 3.29 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.

3.2.4 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.

3.2.4.1 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[3] 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.
- *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.

3.2.4.2 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) [115] and BayesNet (BN) [99]; two Support Vector Machine meth-

[3]Although profile fields *alone* cannot effectively detect crowdturf accounts (§3.2.3), 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 3.7: Classifier configurations.

ods [176]: SVM with radial basis function kernel (SVMr) and SVM with polynomial kernel (SVMp); and two Tree-based methods: C4.5 Decision Tree (J48[4]) [178] and Random Forests (RF) [55]. We leverage existing implementations of these algorithms in WEKA [94] 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 3.7. 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.[5] Figure 3.30 presents their classification error rates, including false positives (classifying normal users as crowdturf workers) and false negatives (classifying crowdturf accounts as normal users).

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

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

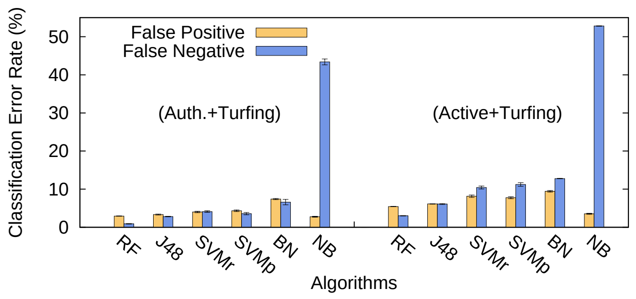


Figure 3.30: Classification error rates. Tree-based algorithms and SVMs outperform Bayesian methods.

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) [60]. 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 [55]. 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.

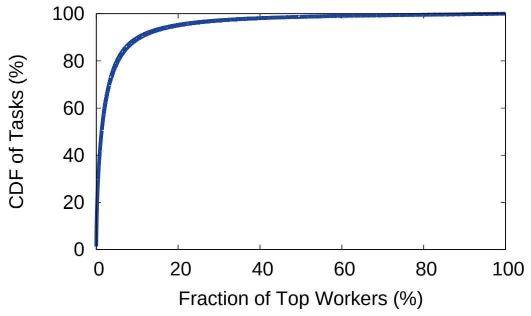


Figure 3.31: % of Tasks finished by top % of workers. The majority of spams were produced by top active workers.

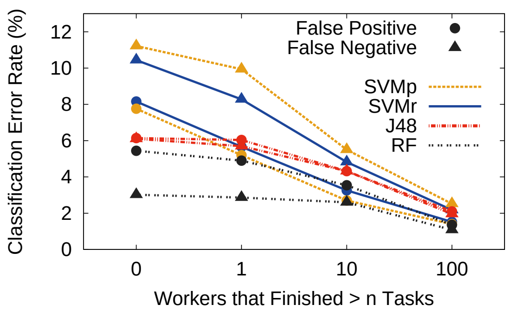


Figure 3.32: Classifying different levels of workers. Workers are filtered by # of crowdturfing tasks finished.

3.2.4.3 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 3.31). Intuitively, these “professional workers” are easier to detect than one-time workers. By focusing 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 3.32 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

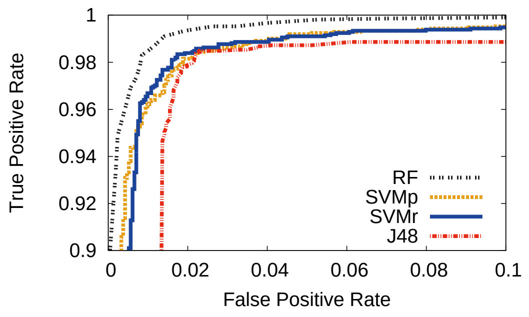


Figure 3.33: ROC curves of classifying professional workers (workers who finished more than 100 tasks).

only 8.9% of the worker population, they are responsible for completing 90% of all tasks. In the rest of our analysis, 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 3.33 shows the ROC[6] 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*

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

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.

3.2.5 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 §3.2.6. First, we define the evasion attack model. We then implement evasion attacks of different strengths, and study the performance 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.

3.2.5.1 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.[7]

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 [46, 213, 241], 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.

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

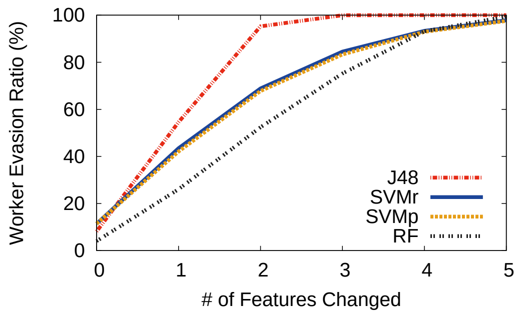
3.2.5.2 Optimal Evasion Attack

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.

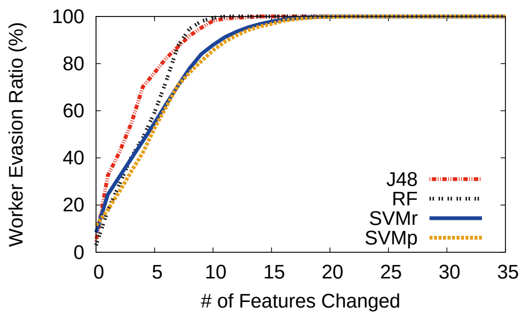
Per-worker Optimal Evasion. Intuitively, each worker should have her own optimal strategy to alter features, *e.g.*some workers need to add followers first, while others need to reduce tweeting burstiness. Doing so is hard in practice: each worker has to apply exhaustive search to identify its optimal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

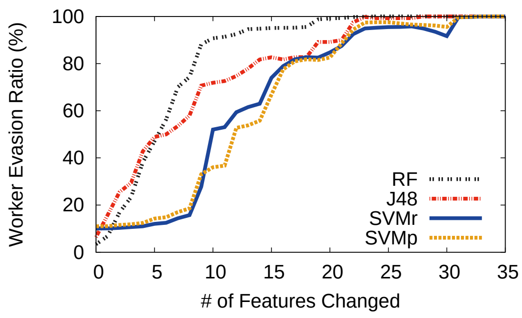
Figure 3.34a plots the evasion rate for the four ML algorithms. Clearly, this optimal evasion model is highly effective. By simply altering one feature, 20-50% of workers can evade detection (different workers can choose to alter different features). And by altering five features, 99% of workers can 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.



(a) Per-worker Optimal Evasion



(b) Global Optimal Evasion



(c) Feature Importance Aware Evasion

Figure 3.34: Evasion rate of optimal evasion strategies (all workers).

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 3.34b. 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.

We implement this attack by building the classifiers and then computing the feature importance. For this we use the χ^2 (Chi Squared) statistic [242], a classic metric to measure feature's discriminative power in separating data instances of different classes[8]. During detection, workers alter features based on their rank.

Figure 3.34c 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 3.34a), 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.

3.2.5.3 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 parame-

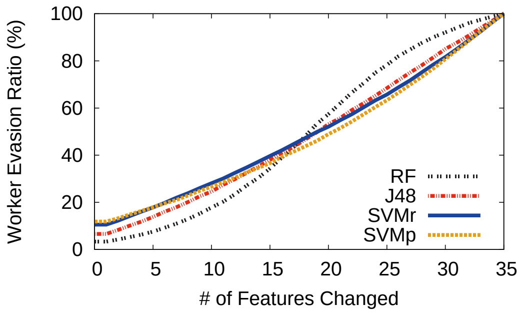
[8]We also tested information gain to rank features, which produced similar ranking results (*i.e.* the same top-10 as using χ^2).

ters) 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.

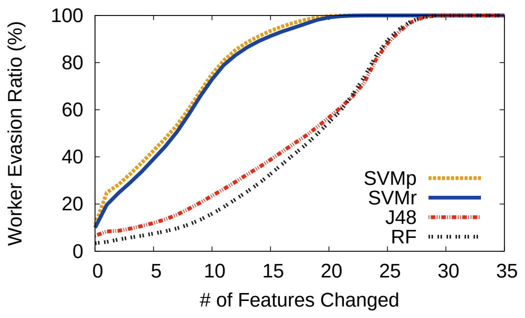
Next, we will examine each constraint separately, and evaluate the likely effectiveness of attacks under the more realistic conditions.

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.

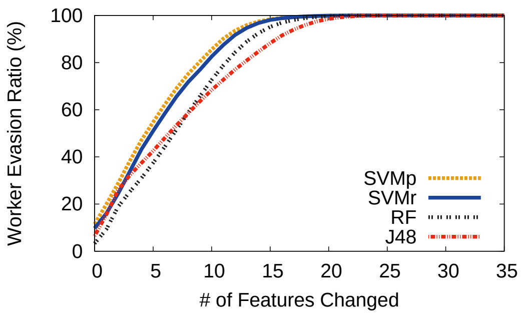
- *Value Distance (VD):* Given a feature k, this captures the distance between a crowd-worker i and normal user statistics by $VD(i,k) = \frac{|F_k(i)-Median(N_k)|}{Max(N_k)-Min(N_k)}$ where $F_k(i)$ is the value of feature k in worker i, and N_k is normal user statistical distribution on feature k. 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.



(a) Random Evasion Strategy (Random)



(b) Value Distance Aware Strategy (VD)



(c) Distribution Distance Aware Strategy (DD)

Figure 3.35: Evasion rate of practical evasion strategies (all workers).

However, our data suggests that this rarely happens (<1%).

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 3.35b and 3.35c show evasion rates based on VD and DD respectively. As a baseline, we also show Figure 3.35a 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

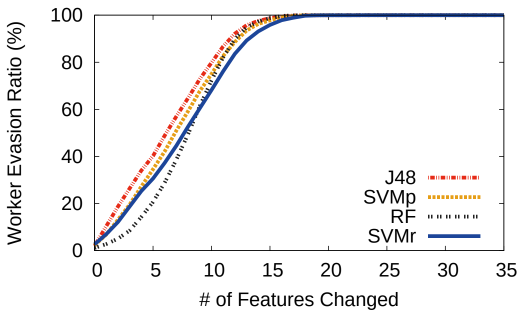


Figure 3.36: Evasion rate using distribution distance aware strategy (DD) for professional workers.

cases (Figure 3.34a–3.34b), 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 3.36. We note that it is easier to evade classifiers dedicated to detect professionals (compared with Figure 3.35c). 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

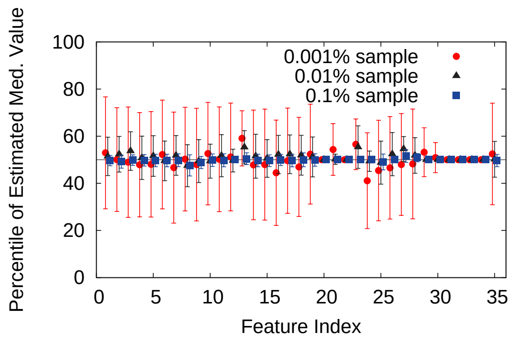


Figure 3.37: The percentile of *estimated* median value in the true normal user CDF.

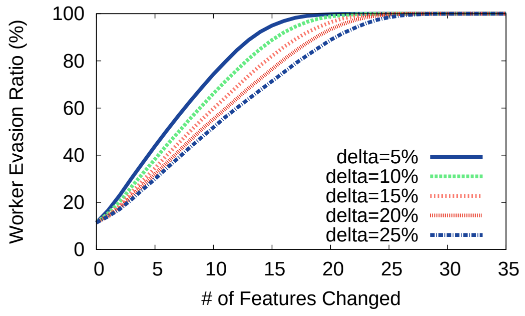


Figure 3.38: Impact of median value estimation error on evasion rate, using DD evasion on SVMp.

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 3.37). 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 3.38). 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.

3.2.6 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.

3.2.6.1 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.

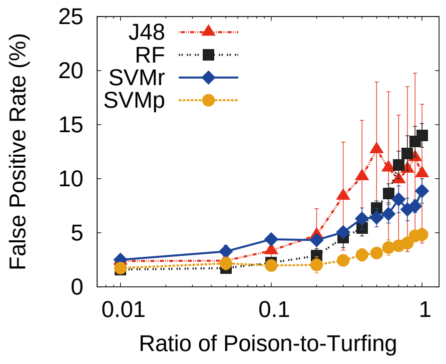
Poisoning Turfing Class. To poison the turfing class, adversaries (*e.g.*ZBJ and SDH administrators) add normal Weibo accounts to the public submission records in their own systems. Since ML classifiers take ground-truth crowdturf accounts from those public records, these benign accounts will then be 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.

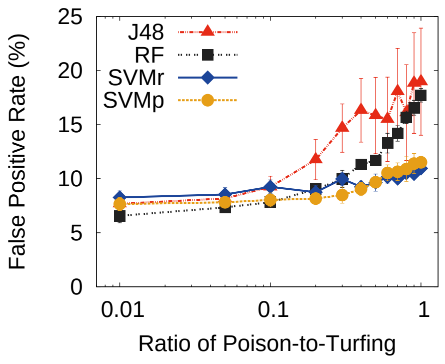
Random Poisoning: We simulate this poisoning attack with *Active+Turfing* dataset, where adversaries inject random normal accounts to the turfing class. Specifically, for training, the turfing class (14K accounts) is a mixture of crowdturf accounts and poison samples randomly selected from *Active*, 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 3.39b. As a baseline comparison, we also present the results of the classifiers for professional workers in Figure 3.39a. 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.[9] Second, we find that the SVM classifiers are more resilient: SVMp's false positive rate increases <5%

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



(a) Professional Workers



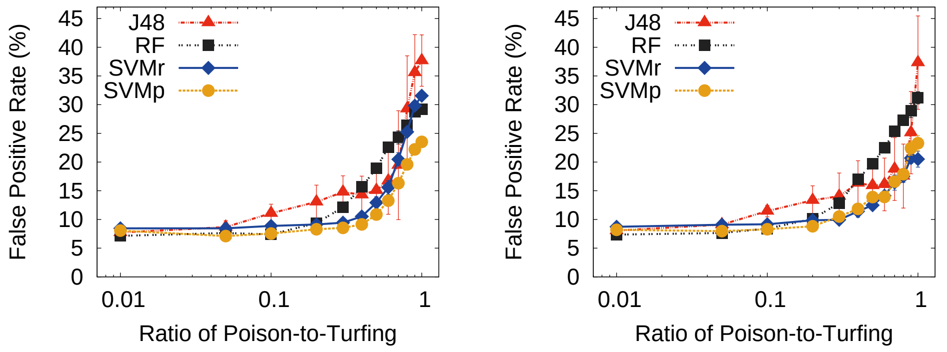
(b) All Workers

Figure 3.39: Poisoning training dataset by injecting random normal user samples to the turfing class.

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 detection of the more homogeneous population of professional workers (§3.2.4) 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 (§3.2.3). 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



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

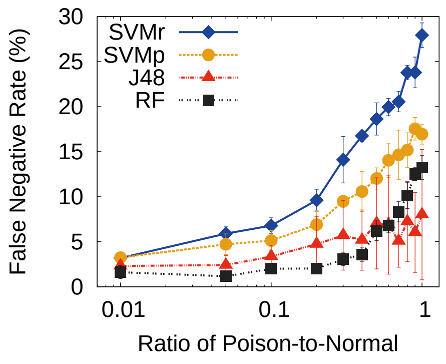
Figure 3.40: Targeted poisoning. Adversaries inject specific type of normal users to the turfing class (all workers).

be malicious, thus are likely to misclassify this kind of normal users as crowd-workers.

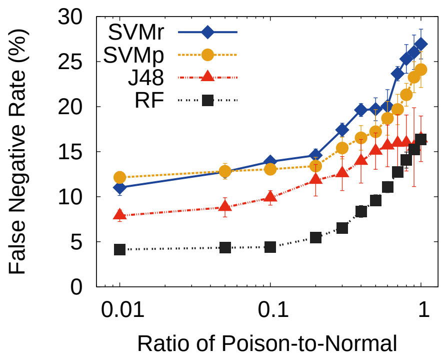
To evaluate the impact of targeted poisoning, we perform similar experiments, except that we select poison samples based on specific feature. Figure 3.40 shows the attacking results on two example features: ratio of tweets with comments and follower count. Compared with Figure 3.39, targeted poisoning 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 3.41a and Figure 3.41b 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 classi-



(a) Professional Workers



(b) All Workers

Figure 3.41: Poisoning training dataset by adding turfing samples to normal class.

fiers 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.

3.2.6.2 Altering Training Data

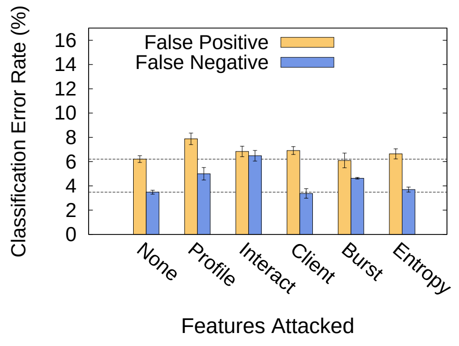
The above poisoning attacks focus on misleading classifiers to catch the wrong target. However, it does not fundamentally prevent crowd-workers from detection, since workers' behavior patterns are still very differently from normal users. To this end, we explore a second poisoning attack, where adversaries 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.

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.

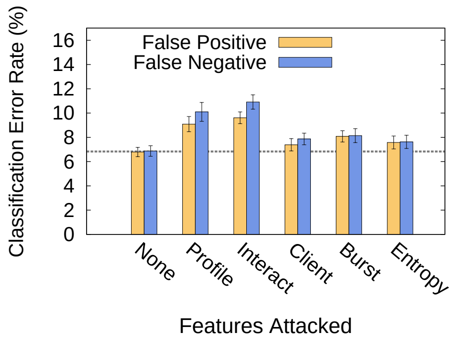
Feature Altering Attack. To simulate this attack, we let adversaries select a set of features F of crowdturf accounts and alter F to mimic the corresponding features of normal users. Unlike evasion attacks that can simply mimic normal users' median values, here we need to mimic the whole distribution 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 3.42 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 3.40). 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.

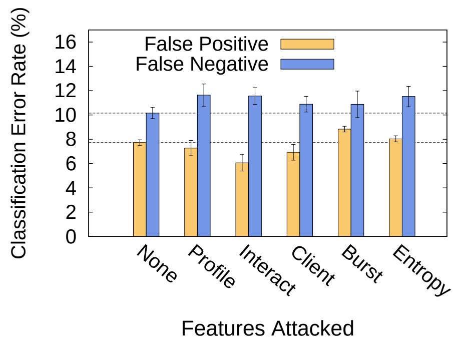
Costs of Altering. In practice, feature altering comes with costs, and some features may be impossible to manipulate even by crowdturfing administrators. For instance, *Tweeting Regularity* (Entropy) and *Burstiness* features are easier to alter. Recall that crowdturfing systems can enforce minimal (random) time delay between workers taking on new tasks, or use delays



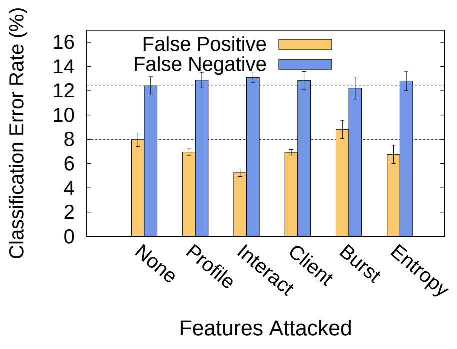
(a) Random Forests



(b) J48



(c) SVMr



(d) SVMp

Figure 3.42: 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 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

Features	Error Rate (FP %, FN %)			
Attacked	**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 3.8: Error rates when features are altered in combinations. We focus on attacking low-cost features: Tweet Client (C), Burstiness (B) and Entropy (E).

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 3.8, 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).

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[10]. 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.

[10]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.

3.2.7 Summary of Results

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.

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 classifiers, 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.

Chapter 4

User Behavior Modeling for Security Defense

So far in Chapter 2 and Chapter 3 we have shown that the next generation of Internet services is increasingly dependent on human users and their content, which makes it more challenging than ever to secure online services. In this chapter, we move our attention to developing practical solutions towards identifying and understanding (malicious) user behaviors in online communities. We build a novel framework for behavior modeling using (semi-)unsupervised learning techniques based on clickstream traces.

In the following, we first explain the clickstream-based user behavior model and the example application for Sybil detection (Section 4.1). Then we extend this to identify and understand more fine-grained user behavior groups within user population (Section 4.2).

4.1 Sybil Detection using Clickstream Analysis

4.1.1 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 [71]. Current defense mechanisms are largely ineffective. Online Turing tests such as CAPTCHAs are routinely solved by dedicated workers for pennies per request [156], and even complex human-based tasks can be overcome by a growing community of malicious crowdsourcing services [157, 230]. The result of this trend is a dramatic rise in forged and malicious online content such as fake reviews on Yelp [219], malware and spam on social networks [77, 88, 213], and large, Sybil-based political lobbying efforts [187].

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 [221] that can be detected using graph theoretic approaches [69, 217, 246, 247]. However, the efficacy of these approaches in practice is unclear. While some Sybil communities have been located in the Spanish Tuenti network [58], another study on the Chinese Renren network shows the large majority of Sybils actively and successfully integrating themselves into real user communities [244].

In this chapter, 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 [91, 139, 166, 189]. 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 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 4.1.2 by describing the problem context and our ground-truth dataset, followed by preliminary analysis results in Section 4.1.3. Next, in Section 4.1.4 we propose our clickstream models to effectively distinguish Sybil with normal users. Then in Section 4.1.5, we develop an incremental Sybil detector that can scale with today's large social networks. We then extend this detector in Section 4.1.6 by proposing an unsupervised Sybil detector, where only a minimal (and fixed) amount of ground-truth is needed. Finally, in Section 4.1.7, we describe experimental experience of testing our prototype code in real-world social networks (Renren and LinkedIn). We then conclude in Section 4.1.8.

4.1.2 Background: Renren and Clickstream

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 [112]. 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 [244]. 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 [244]. 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. 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 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.

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 4.1: Clickstream dataset.

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 4.1. 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 4.1.3.2.

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 [47]. 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.

4.1.3 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

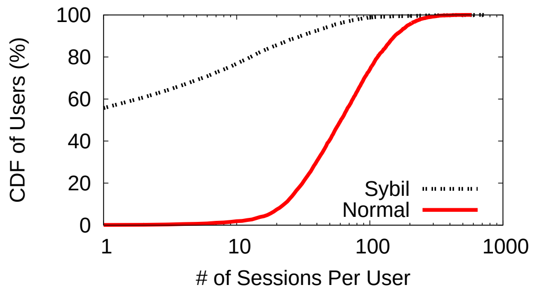


Figure 4.1: # of sessions per user.

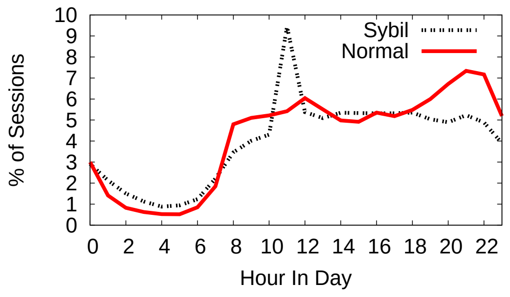


Figure 4.2: Sessions through the day.

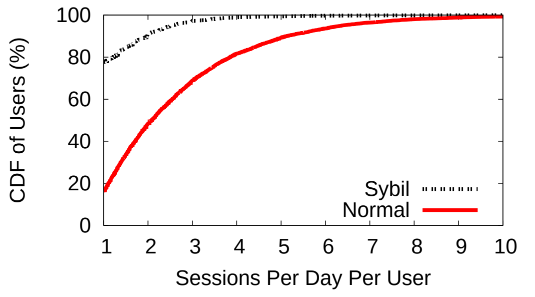


Figure 4.3: Sessions per day per user.

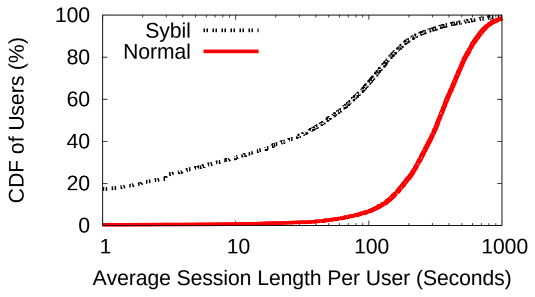


Figure 4.4: Average session length per user.

use a Support Vector Machine (SVM) approach to learn the important features that distinguish Sybil and normal user clickstreams.

4.1.3.1 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 4.1, >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 4.2 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.

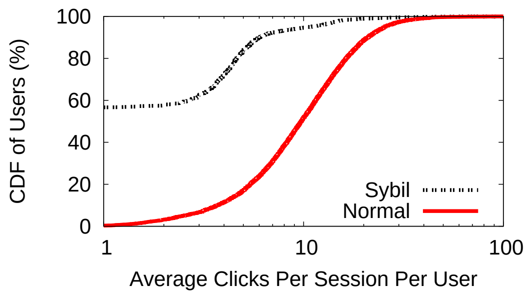


Figure 4.5: Average # of clicks per session per user.

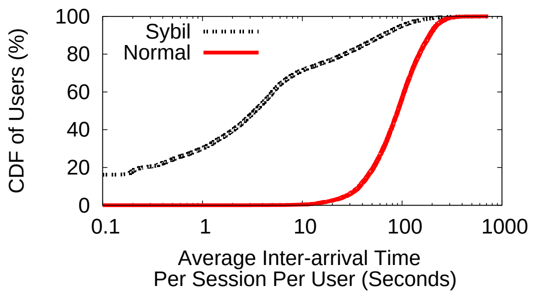


Figure 4.6: Average time interval between clicks per session per user.

Next, we investigate the number of sessions per user per day. Figure 4.3 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 4.4: 70% of Sybil session are <100 seconds long, versus 10% of normal sessions. The vast majority of normal sessions last several minutes.

Figure 4.5 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 ≥ 10 clicks. Not only do Sybil sessions tend to be shorter, but Sybils also click much faster than normal users. As shown in Figure 4.6, 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 4.2, Sybils spend a great deal of clicks sending friend requests and browsing profiles, despite Renren's security restrictions.

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 4.2: 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.

4.1.3.2 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 4.2, 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.
- **Like:** Corresponds to the user liking (or unliking) content on Renren.

Table 4.2 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 4.2 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 [109], passively collecting friends [213]), 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

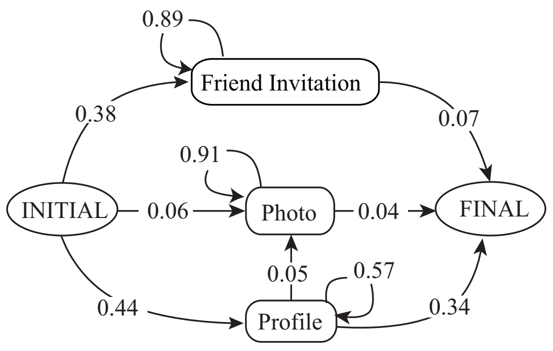


Figure 4.7: State transitions for a Sybil account.

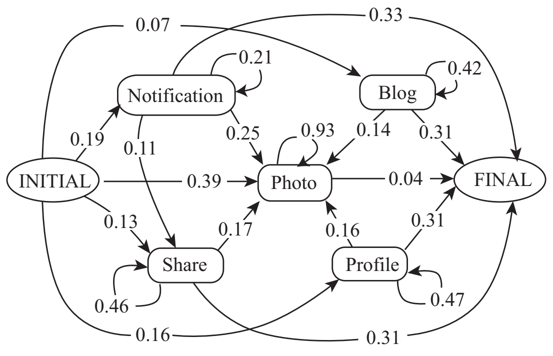


Figure 4.8: State transitions for a real user.

illegitimate sharing/messaging indicate that spam detection systems cannot simply leverage numeric thresholds to detect spam content.

4.1.3.3 Click Transitions

Sections 4.1.3.1 and 4.1.3.2 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 4.7 and Figure 4.8 show 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 4.7 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 4.8) 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 [236, 112].

It is clear 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.

4.1.3.4 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) [177]. 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 4.1.3.2, there are 8 categories of clicks activities on Renren. For each user, we use the percentage of clicks in each category as a

Feature	**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 4.3: Weight of features generated by SVM.

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 4.3 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.

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.

4.1.3.5 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.

4.1.4 Clickstream Modeling and Clustering

In Section 4.1.3, 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.

4.1.4.1 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 4.1.3, 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 4.6, 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 4.1.3.2). Our experimental analysis evaluates both representations to understand the impact of granularity on model accuracy.

4.1.4.2 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.

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 consecutive elements) in sequence S: $T_k(S) = \{k$-

$gram|k\text{-}gram = (s_i s_{i+1} ... s_{i+k-1}), i \in [1, n+1-k]\}$. 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* [135], we define the distance between sequences S_1 and S_2 as:

$$D_k(S_1, S_2) = 1 - \frac{|T_k(S_1) \cap T_k(S_2)|}{|T_k(S_1) \cup T_k(S_2)|} \tag{4.1}$$

We will discuss the choice of k in Section 4.1.4.2.

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:

$$D(S_1, S_2) = \frac{1}{\sqrt{2}} \sqrt{\sum_{j=1}^{n} (c_{1j} - c_{2j})^2} \tag{4.2}$$

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:

$$D(S_1, S_2) = sup_t|F_{n,1}(t) - F_{n',2}(t)| \tag{4.3}$$

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

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 4.4 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.

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 4.4, 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 Hy-

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

Table 4.4: Summary of distance functions.

brid 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 4.6, 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 4.4, 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.

4.1.4.3 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 4.1.3, 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 [121], 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 4.1.4.4.

Cluster Quality. A key question when evaluating our methodology is assessing the quality of clusters produced by METIS. In Section 4.1.4.4, 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.

4.1.4.4 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

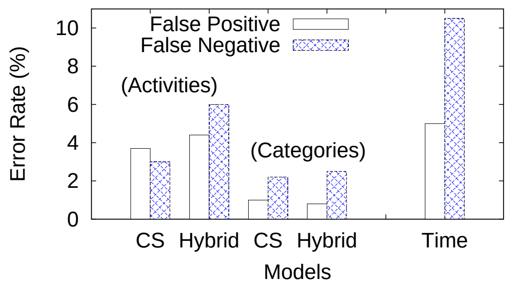


Figure 4.9: Error rate of three models.

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 4.9. 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 4.9. 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.

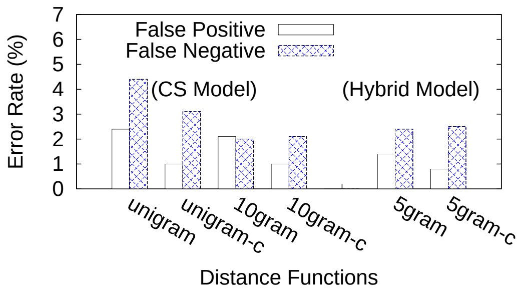


Figure 4.10: Error rate using different distance functions.

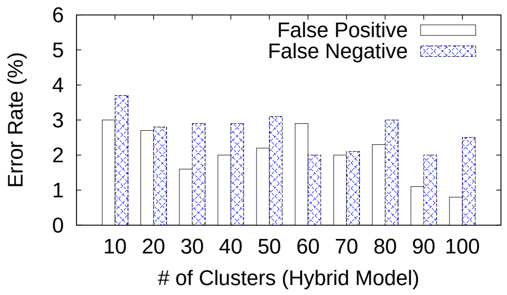


Figure 4.11: Impact of number of clusters (K).

The second conclusion from Figure 4.9 is that encoding clicks based on category outperforms encoding by activity. This result confirms findings from the existing literatures on web usage mining [44]: 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 analysis, 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 4.10, 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 4.10. 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 4.11 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 4.1.6, 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.

4.1.5 Incremental Sybil Detection

Our results in Section 4.1.4 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 4.1.4. 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.

4.1.5.1 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.

4.1.5.2 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

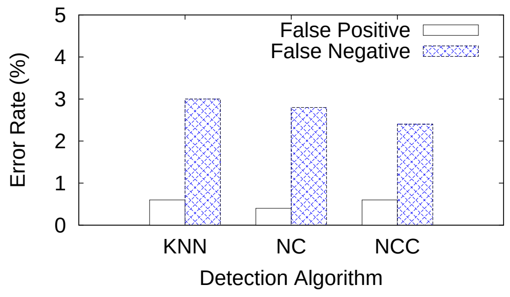


Figure 4.12: Error rate of three reclustering algorithms.

Sybils and 3000 normal users. We build sequence similarity graphs from the training data 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 4.12, 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 4.13 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

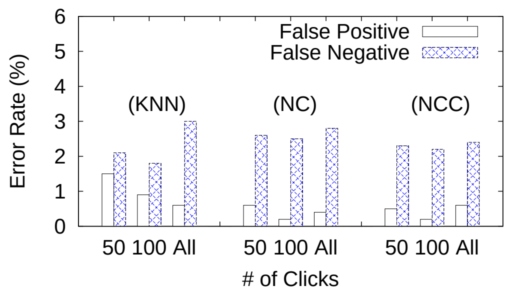


Figure 4.13: Error rate vs. maximum # of clicks in each sequence.

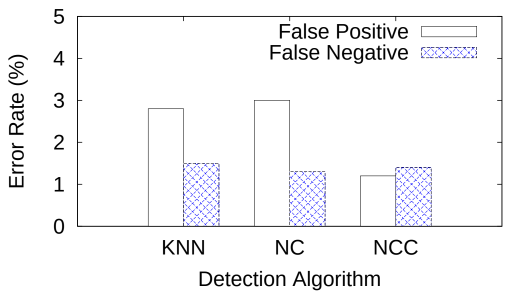


Figure 4.14: Detection accuracy when training data is two weeks old.

length.

Surprisingly, the “All” results are not the most accurate overall. As shown in Figure 4.13, 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 4.13. 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

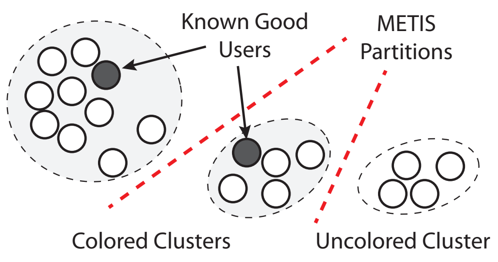


Figure 4.15: Unsupervised clustering with coloring.

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 4.14 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.

4.1.6 Unsupervised Sybil Detection

Our incremental Sybil detection system from Section 4.1.5 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 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 4.15 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 4.1.5.

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.

4.1.6.1 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

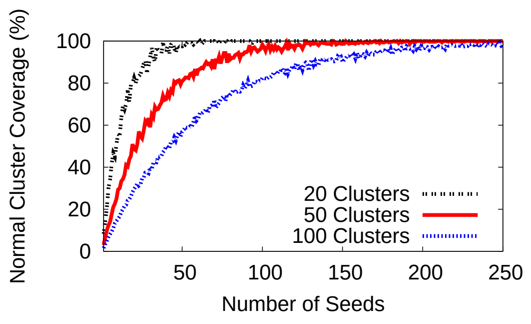


Figure 4.16: # of seeds vs. % of correctly colored normal user clusters.

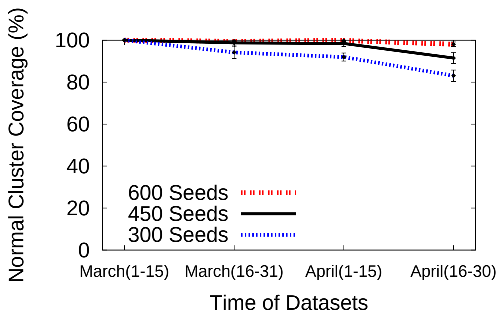


Figure 4.17: Consistency over time of normal seeds for coloring.

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 4.1.5, 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 4.16 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 4.17. 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.

4.1.6.2 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 4.1.5, *i.e.*by comparing the composition of the colored clusters to ground-truth.

The results are shown in Figure 4.18. 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 tradeoff. 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

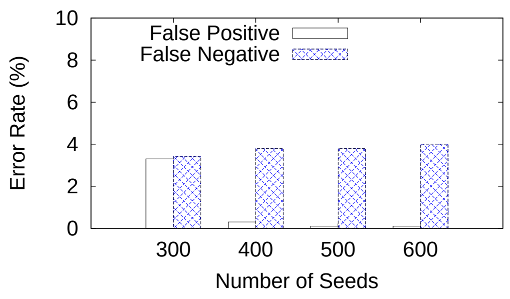


Figure 4.18: Detection accuracy versus number of seeds.

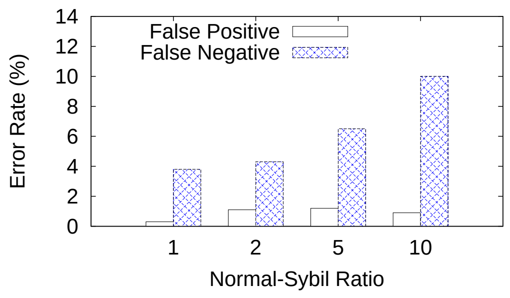


Figure 4.19: Detection accuracy versus Normal-Sybil ratio.

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 [12].

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 4.1.5.1). We ran additional tests using the NC and KNN algorithms, but the results were very similar and we omit them for brevity.

Figure4.19 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 [213, 244], systems that verify user real-world identity [22], or even Sybil detection systems using crowdsourced human inspection [225]. These efforts could further protect benign users from misclassification.

4.1.7 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.

4.1.7.1 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.

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 [46, 77, 222, 225, 245].

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.

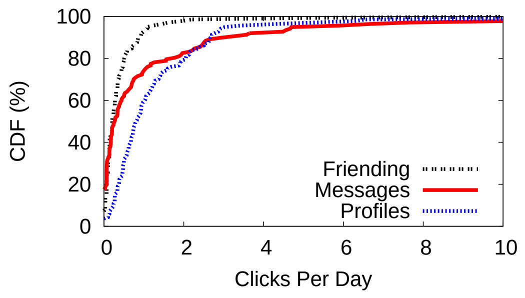


Figure 4.20: Clicks per day by outlier normal users.

4.1.7.2 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 4.20. 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 4.20, 99% of outliers generate $\leq$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.

4.1.8 Summary of Results

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.

4.2 Interpretable User Behavior Model

4.2.1 Introduction

Thus far, we have demonstrated the effectiveness of using clickstream similarity graph to detect Sybils. However, the current model only performs binary classification among users (*i.e.*, either malicious or benign). It helps to capture attackers but does not provide explicit knowledge about how users (or attackers) behave and how their behavior changes over time. For instance, among the Sybil accounts, there are likely different attacking strategies used by different attackers while the current model cannot differentiate them. Even among real users, there can be undesired behaviors such as bully or trolling that are remain undetected. So in the following sections, we extend this clickstream model beyond binary classification to identifying and understanding fine-grained user behaviors in online services.

Understanding user behavior in today's online services is a complex and difficult challenge. In systems with millions of users, how can system builders understand the factors that drive each user's behavior? Understanding such factors can dramatically improve a user's experience, either through better performance, customized user interface features, or better targeted ads. Take for example the LinkedIn social network. LinkedIn is used by different types of users ranging from students not yet on the job market, happily employed professionals, professionals seeking new positions, and recruiters. Each user type uses the service differently, and yet rarely identifies their usage type explicitly in their profile data or elsewhere.

The intuitive solution is to survey users on how they use these systems through well-designed user studies [49, 243]. Unfortunately, this approach is limited by three factors. First, detailed user studies are limited in scale because of their significant cost to conduct and analyze. Studies sacrifice scale for depth on a small sample of the user population. Second, users may not be willing or able to self-identify into different user categories. Finally, user surveys rely on known questions or hypotheses. Unknown or new user behaviors cannot be anticipated in these studies.

These issues can be addressed by a data-driven approach to understanding user behavior. With improving data mining tools, today's online services collect all traces of user activity to produce *clickstreams*, sequences of timestamped events generated by user actions. For web-based services, these might include detailed HTTP requests. For mobile apps, clickstreams can include everything from button clicks, to finger swipes and text or voice input. Compared to user studies, clickstream analysis can scale to large user populations, identify behaviors without user assistance, and identify previously unknown behaviors.

Yet identifying common user behaviors in clickstreams is very challenging. Early works on clickstreams are limited, and focused on users' "navigation paths" within a website [200, 216, 100], or use Markov Chain models to predict popular webpages [189, 139]. To identify user behaviors today, we need a sophisticated clickstream analysis system that meets three requirements. *First*, it must scale and function well on large, noisy clickstream datasets. *Second*, the system should be able to capture previously unknown user behavior, *i.e.* capture behavior without categories or labels defined a priori. This is critical, because users often utilize popular services in unexpected ways, and adapting to these behaviors can determine the long-term viability of a service. *Finally*, the system should be interactive, and help others understand user behavior by presenting detected behaviors in an intuitive and understandable way. In contrast, current tools usually treat user models as a "black box" for classification tasks, while offering little explanations on how users behave and why [91].

In this work, we present the design and evaluation of a practical and scalable clickstream tool for user behavior analysis. Based on clickstream similarity graph (previous section), we use a hierarchical clustering approach to detect the most popular behavior patterns, and use an *iterative feature pruning* technique to remove the influence of dominating features from each subsequent layer of clusters. The result is a hierarchy of behavioral clusters where higher-level clusters represent more general user behavior patterns, and lower-level clusters further identifying smaller groups that differ in key behavioral patterns. We can further use Chi-square statistics to identify statistical features that can be used to categorize and label behavior clusters.

Our system provides an easy way for service providers to analyze and understand groups and patterns in user behavior. First, the hierarchy of behavior clusters presents a compressed view of the most dominant user behavior patterns. In addition, because our approach does not rely on prior knowledge of categories or labels, it is able to capture any behavior patterns, both known and unknown. Finally, we integrate an *interactive visualization tool* to help service providers to examine the clustering results in real time.

To demonstrate the effectiveness of our system, we perform case studies using two large-scale, real-world clickstream datasets. One clickstream captures 135 million smartphone app events from 100K users on Whisper, a popular anonymous social network app. A second dataset comes from Renren (China's Facebook) and contains 7 million click events from 16K normal and malicious users. Our tool produces user behavioral models and reveals key insights about users on both networks. First, we identify patterns that capture different levels of "dormant users" on Whisper, and effectively predict dormant users based on neighboring behavior clusters. Second, we study user blocking behavior on Whisper and show that much of the blocking behavior is bidirectional, often following private message sessions, and is often correlated with sexually suggestive messages (sexting). On our Renren dataset, our system not only accurately identifies fake accounts with 95% accuracy, but also reveals subgroups that utilize different attack strategies. For example, we identify attacker subgroups that try to emulate normal users by intentionally slowing down their attack speed to avoid detection.

Finally, we evaluate our tool on two key benchmarks: First, we evaluate whether the algorithm-generated behavioral cluster are easy to understand with a controlled user study. We let participants summarize the corresponding user behaviors in a given cluster by examining cluster features. We find that most participants can interpret the semantic meaning of the user behavior and their summaries reach a high level of *consistency*. Second, we evaluate the clustering quality produced by our algorithm, in comparison to existing clustering methods (*e.g.*, K-means). Results show that our approach reaches a higher *accuracy* in detecting and grouping similar users.

We make three key contributions.

- We propose a novel unsupervised method to model online user behaviors. By building and partitioning a clickstream similarity graph, we capture the detailed user behavior models as hierarchical clusters in the graph. In addition, our tool automatically produces intuitive features to interpret the meaning of the behavioral clusters.
- We perform real-world case studies on two large-scale clickstream traces (142 million click events in total). We demonstrate that our tool can effectively help service providers to identify unexpected user behaviors (malicious accounts in Renren, hostile chatters in Whisper) and even predict users' future actions (dormant users in Whisper).
- Finally, we perform benchmark evaluations on our tool. The results show that the algorithm-generated cluster labels are easy to understand, and our tool produces highly accurate user behavioral models.

4.2.2 Whisper Datasets

In this work, we seek to build a clickstream tool for user behavioral modeling in online services. To provide context, we first describe the clickstream datasets used in our study. In addition to the Renren clickstream dataset described in the above section (Section 4.1), we introduce a new clickstream dataset from a popular mobile social network Whisper. In the following, we describe Whisper and the clickstream dataset in details. Note that we have taken careful precautions to avoid any personally identifiable information in our datasets, and our study has been approved by our local IRB under protocol #COMS-ZH-YA-010-6N.

Whisper is a popular smartphone app for anonymous social messaging. It allows users to share confessions and secrets under anonymous nicknames without worrying about privacy [66]. As of April 2015, Whisper has reached 10 million users. Unlike traditional social networks, Whisper does not maintain user profiles or social connections. Its key function is messaging: the app overlays a user's short text message on top of a background picture se-

Category	Event Type	Events # (K)	Events %	Initiated By User?
Browsing	View whisper	52437	38	Yes
	View popular feed	16008	12	Yes
	View nearby feed	5354	4	Yes
	View latest feed	2346	2	Yes
	View other feed	196	1	Yes
Account	Login	16994	12	Yes
Posting	Heart whisper	2156	2	Yes
	Upload image	1325	1	Yes
	Create whisper	1308	1	Yes
Chatting	Being blocked in chat	3271	3	No
	Block user in chat	3271	3	Yes
	Start a chat	2238	2	Yes
Notification	Receive notification	9680	7	No
	Whisper recommendation	2530	2	No

Table 4.5: Event types in the Whisper dataset. # of click events are presented in thousands. Events that are <1% are omitted for brevity.

lected by keywords. The resulting *whisper* message is posted to the public stream where other users can read, reply or heart (like) it. In addition, the app provides a chat feature to facilitate direct communication. Any user can start a private chat with the whisper author. Finally, users browse whispers from several public lists.

We collect detailed clickstream data from Whisper in collaboration with Whisper's Data Science team. The dataset contains 135,208,159 click events from 99,990 users over 45 days in 2014. Users were randomly selected from the Whisper user population as a representative sample. Each click event is characterized by userID, timestamp, event type and event parameter. The userID in our dataset (including Renren data) is globally unique and has been fully anonymized to protect user privacy. We obtained userIDs from each company through internal collaborators. The Whisper dataset contains 33 types of events that can be grouped into 6 categories. These categories are:

- **Browsing:** Browsing whispers, visiting the public whisper feeds (popular/nearby/latest list).

- **Account:** Creating a user account and login the app.
- **Posting:** Posting original whispers and replies, hearting/unhearting a whisper, sharing whispers, and tagging a whisper to a topic.
- **Chatting:** Initiating a chat, blocking other users in a chat, and being blocked in a chat.
- **Notification:** Receiving notifications about hearts/replies on their whispers, and whisper recommendations.
- **Spam:** Whisper being examined or deleted by system admins, flagging other people' whispers. Events in this category are all below 1% (omitted from Table 4.5).

Among the 33 event types, 25 are user-initiated events corresponding to the user performing an action on the app (*e.g.*, "posting a whisper"). The rest 8 events are system events which don't require user action (*e.g.*, "receiving notifications"). Table 4.5 shows the most popular events and the absolute number (in thousands) and the percent of clicks. Overall, the most prevalent events are related to content consumption such as viewing whispers. Interestingly, under the chatting category, the most prevalent events are "blocking users" and "being-blocked" by others. Intuitively, anonymous environment is more likely to foster abusive behaviors (*e.g.*, bullying) [210]. Later, we investigate this behavior in greater details using behavioral models.

Our dataset also contains the content of the public whispers (about 1 million) posted by these users. This content data is not used to construct clickstreams, but used to understand specific user behavior and user intent later in our analysis.

4.2.3 Unsupervised User Behavior Modeling

In this section, we describe our unsupervised method to build user behavior models from clickstream data. At the high level, our system assumes that human behavior naturally forms clusters. Despite users' differences in personalities and habits, their behavioral patterns within

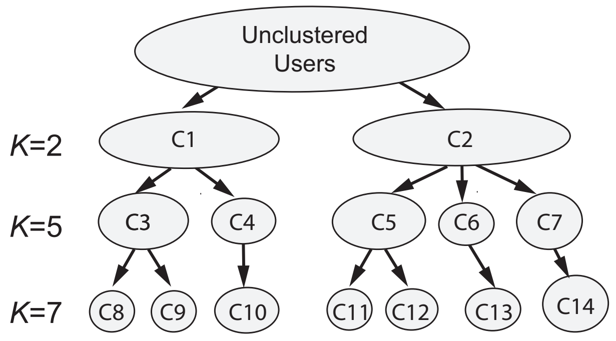


Figure 4.21: Hierarchy of the behavioral clusters.

a given service cannot be entirely disparate. Our goal is to identify such natural clusters as behavioral models. In addition, user behavior is likely multi-dimensional. We expect user clusters to fall into a tree hierarchy instead of a one-dimensional structure (Figure 4.21). In this hierarchy, most prominent features are used to place users into high-level categories while less significant features characterize detailed sub-structures.

To these ends, we design an algorithm to captures hierarchical clickstream clusters with *iterative feature pruning*. At the high-level, we partition the similarity graph to identify clusters of users with similar clickstream activities. To capture the hierarchical structure, we recursively partition newly generated clusters, while *pruning* the feature set used to measure clickstream similarity. Intuitively, by identifying and pruning dominating features in higher-level clusters, we allow the secondary features to manifest and discover more fine-grained subclusters. Also, the pruned features are indicative of why this cluster is formed, which can help service providers to understand the behavioral model.

In the following, we describe the feature-pruning algorithm to identify clusters in the similarity graph. Finally, we build a visualization tool to help service providers examine and understand behavioral clusters.

4.2.3.1 Feature Pruning based Clickstream Clustering

A similarity graph dominated by very few features gives little insight on subtle differences between users. The generated clusters may only describe the broadest user categories, while interesting and detailed behavioral patterns remain hidden. We recognize that similarity graph has the capability to capture user behavior at different levels of granularity. We implement *iterative feature pruning* as a means of identifying fine-grained behavioral clusters within existing clusters, and recursively partitioning the similarity graph. In the following, we first introduce the key steps of our clustering algorithm and feature pruning. Then we describe using pruned features to interpret the meaning of the clusters, and the technical details to determine the number of clusters.

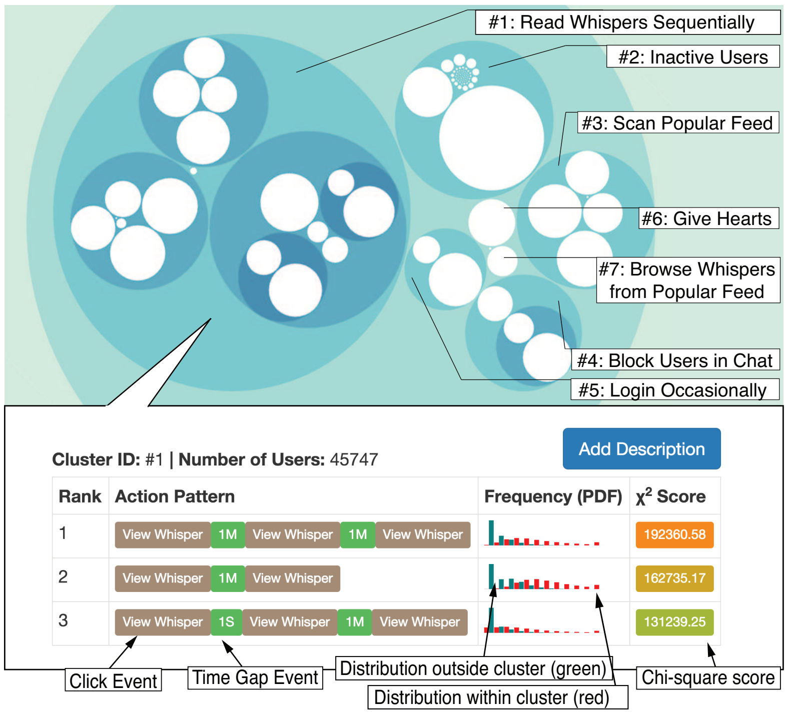
Iterative Feature Pruning & Clustering. We explain how our algorithm works using the example in Figure 4.21. We start with a similarity graph of all users, where clickstream similarity is measured based on the full feature set (union of all k-grams). By partitioning the similarity graph, we get the top-level clusters C_1 and C_2. The partitioning algorithm we use is Divisive Hierarchical Clustering [122], which can work on arbitrary metric space and find clusters of arbitrary shapes.

To identify more fine-grained subclusters within C_1 or C_2, we perform feature pruning: We identify the primary features that are responsible for forming the parent cluster, remove them from the feature set, and use the remaining secondary features to further partition the parent. For example, suppose C_1 is the current parent cluster. We first perform feature selection to determine the key features (*i.e.*, k-grams) that classify users into C_1. Then to partition C_1, we remove those top k-grams from the feature set, and use the remaining k-grams to compute a new similarity graph for C_1. In this way, secondary features can step out to partition C_1 into C_3 and C_4 (by running Divisive Hierarchical Clustering on the new similarity graph). For each of the newly generated clusters (*e.g.*, C_3 and C_4), we recursively run the same process to produce more fine-grained subclusters. Our algorithm stops when all the new partitions cannot be further split, *i.e.* clustering quality reaches a minimal threshold. The result is a tree hierarchy of behavioral clusters.

The key step of feature pruning is finding the primary features responsible for forming the parent cluster. We select features based on Chi-square statistics (χ^2) [242], a classic metric to measure feature's discriminative power in separating data instances of different classes. For a given cluster, *e.g.*, C_1, we measure the χ^2 score for each feature based the distribution of users in C_1 and those outside C_1. We sort and select the top features with the highest χ^2 scores. Our empirical data shows χ^2 distribution usually exhibits "long-tail" property — only a small number of dominating features have high χ^2 scores. We automatically select top features by identifying the sweet point (or turning point) in the χ^2 distribution [190].

Understanding the Behavioral Clusters. We can infer the meaning of the clusters based on the selected features during feature pruning phase. A feature is selected because users in this cluster are distinct from users outside the cluster on this particular feature dimension. Thus it can serve as explanations for why a cluster has formed and which user behaviors the cluster encompasses. Later we construct a visualization tool to help service providers interpret behavioral clusters.

Determining the Number of Subclusters. For each parent cluster (and its similarity graph), our system identifies the natural number of subclusters within. To do so, we monitor the changes of the overall *clustering quality* while continuously partitioning the graph to more subclusters. We stop when generating more subclusters will no longer improve the clustering quality. The metric to assess clustering quality is the widely-used *modularity*, which measures the density of edges inside clusters to edges outside clusters [52]. The modularity value ranges from -1 to 1, with a higher value indicating a better clustering quality.



Cluster ID: #1 | **Number of Users:** 45747

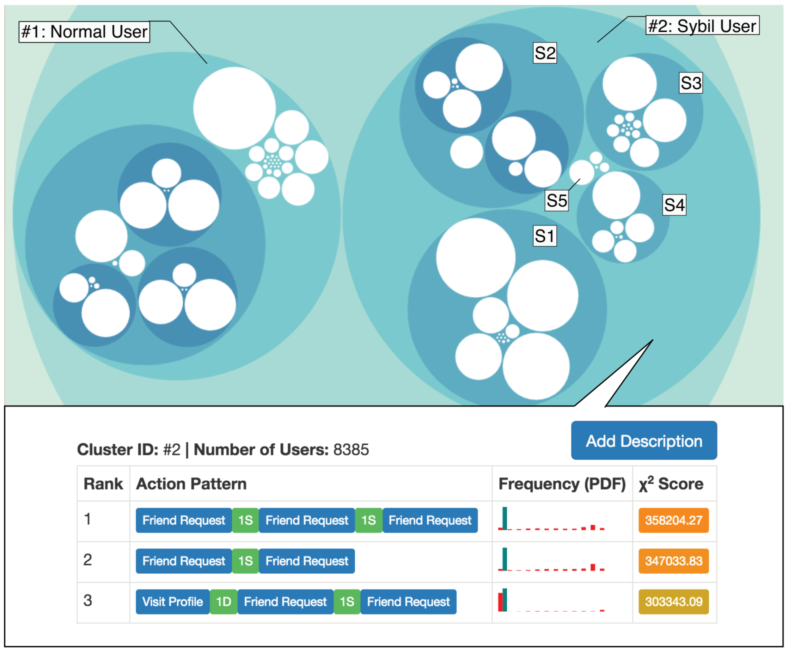
Rank	Action Pattern	Frequency (PDF)	χ^2 Score
1	View Whisper 1M View Whisper 1M View Whisper		192360.58
2	View Whisper 1M View Whisper		162735.17
3	View Whisper 1S View Whisper 1M View Whisper		131239.25

Figure 4.22: Whisper behavioral clusters. Cluster labels are manually input based on results of each cluster. The pop-up window shows users in Cluster #1 tend to sequentially read whispers.

4.2.3.2 Cluster Visualization

We build a visualization tool for service providers to examine and understand user behavioral clusters generated by our algorithm. The tool allows service providers to answer key questions about their users, *e.g.*, what are the major behavioral categories? Which behavior is more prevalent? What's the relationship between different types of behavior?

Visualization Interface. Figure 4.22 shows a screenshot of our tool displaying the behavioral clusters of Whisper (best viewed in color). We build this tool using *D3.js*, a JavaScript library for data visualization. By default, we display the cluster hierarchy using Packed Circle [232], where child clusters are nested within their parent cluster. This gives a clear view of the hierarchical relationships of different clusters. Circle sizes reflect the number of users in the cluster, which allows service providers to quickly identify the most prevalent user behav-



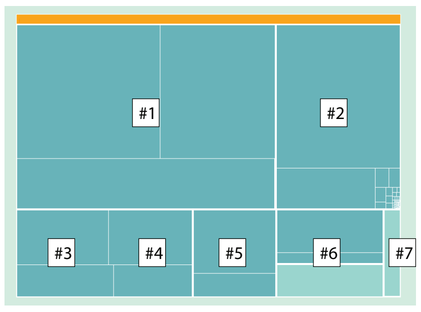
Cluster ID: #2 | **Number of Users:** 8385

Rank	Action Pattern	Frequency (PDF)	χ^2 Score
1	Friend Request 1S Friend Request 1S Friend Request		358204.27
2	Friend Request 1S Friend Request		347033.83
3	Visit Profile 1D Friend Request 1S Friend Request		303343.09

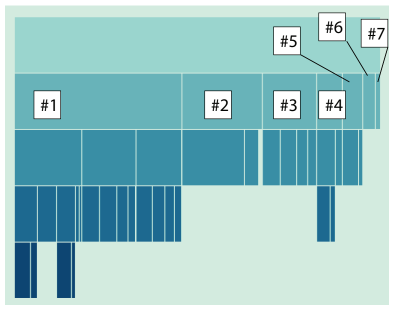
Figure 4.23: Renren behavioral clusters. The pop-up window shows users in Cluster #2 focus on sending friend requests and browsing user profiles.

iors. Finally, the visualization tool is zoomable and easy to navigate among clusters. We also implemented other interfaces such as Treemaps [116], Sunburst [202] and Icicle [127]. Service providers can choose any of these based on personal preference (Figure 4.24). We use Packed Circle as default because it leaves more space between clusters, making it easier to visually separate different clusters.

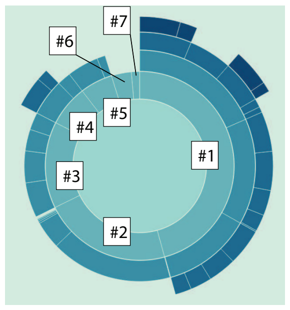
To understand the user behavior in a specific cluster, we can click the cluster to pop-up an information window. Take the one in Figure 4.22 for example: we show the basic cluster information on top, including clusterID and the number of users. Below is a list of "Action Patterns" (k-grams) selected by our Feature Pruning algorithm to describe how users behave. Each row contains one pattern, ranked by χ^2 score (brighter color indicates higher score). The "Frequency (PDF)" column shows how frequently each action pattern appears among users of this cluster. The red bar indicates the pattern frequency (probability density function) inside the cluster, and the green bar denotes frequency outside of this cluster. Intuitively, the more



(a) Treemaps



(b) Icicle



(c) Sunburst

Figure 4.24: Whisper hierarchical clusters displayed with different visualization methods. We mark the cluster number of the top-level clusters in the text box.

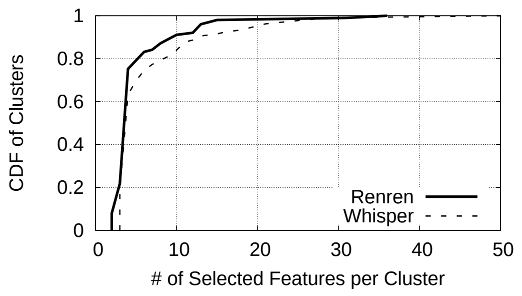


Figure 4.25: # of Selected features per cluster.

divergent the two distributions are, the more distinguishing power the pattern has. In this example, the first pattern shows users viewing whispers sequentially with a time interval of one minute or less. The red bar is much more skewed to the right, indicating users in this cluster perform this action more often than users outside. Finally, service providers can “add descriptions” to the cluster using the button in blue.

Visualizing Whisper and Renren Clusters. We run our system on Whisper and Renren datasets and display the behavioral clusters in Figure 4.22–4.23. We apply the same configuration on both runs: the partitioning of a cluster stops if the modularity reaches a threshold 0.01 (insignificant cluster structure). We intentionally set a loose threshold to let the algorithm dig out very detailed sub-clusters. In practice, service providers can tune this parameter depending on how detailed behavioral clusters they need. For our Whisper dataset, our system produces a tree hierarchy of 107 clusters (root included) with 95 leaf clusters. The maximum tree depth is 4. For Renren, the hierarchy contains 108 clusters (95 leaf clusters) with a maximum depth of 4.

Note that our visualization tool only displays the selected features for each cluster. As shown in Figure 4.25, 80% of the clusters have less than 5 selected features, and 90% of the clusters have less than 10. This indicates that the prevalent user behavior can be characterized by a small number of key feature dimensions. Also, this makes it possible for people to understand the cluster without looking through the full feature set (*e.g.*, Whisper data has 80903 unique kgrams as features).

4.2.4 Evaluation: Cluster Labels

In the following, we analyze the behavioral clusters in Whisper and Renren, and demonstrate their effectiveness in identifying unexpected behavior and predicting future activities. Our evaluation contains three steps. First, To evaluate the ease of understanding and labeling behavioral clusters, we run a user study. We ask the participant to read cluster information and describe the corresponding user behavior. Then we examine whether different people give consistent descriptions. Second, we perform in-depth case studies on the unusual behavioral clusters, and provide new insights to both networks. Third, we evaluate cluster quality, *i.e.*, how well behavioral clusters capture similar users.

4.2.4.1 User Study to Interpret Clusters

User behavioral models need to be intuitive and understandable to the service providers. Thus we conduct a user study to answer two key questions: Are these behavioral clusters understandable to humans? How consistently do different people interpret the corresponding user behaviors?

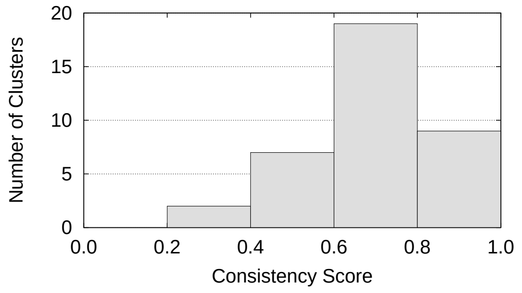


Figure 4.26: Distribution of consistency score.

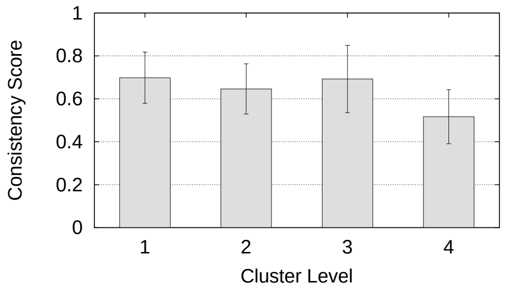


Figure 4.27: Consistency score vs. cluster level.

In this user study, we ask participants to browse behavioral clusters using our visualization tool (Packed Circle interface). For each cluster, the participant is asked to describe the user behavior using her own words (in one sentence) based on the information displayed. If a cluster is not understandable to the participant, she can mark it as “N/A”. Since our tool is designed for service providers, we expect they will have basic technical backgrounds. Our participants include 15 graduate students in Computer Science who have basic knowledge in online social networks. To best utilize participants’ time, we only use the Whisper clusters (Figure 4.22), and the participants only look at top clusters that cover 90% of users at each level of the hierarchy (37 clusters in total). Before the test, we ask the participants to use the Whisper app for at least 10 minutes to get familiar with it. Each participant also goes through a quick instruction session to learn how to use the visualization tool and how to read the information in the pop-up window.

4.2.4.2 User Study Results

We gathered a total of 555 descriptions from the participants on the 37 clusters (15 descriptions per cluster). We find that the behavioral clusters are generally understandable to the participants. Out of the 555 descriptions, 530 (95.5%) are valid descriptions about user behaviors (others are “N/A” marks). In addition, most participants can finish the task within a reasonable amount of time. The average completion time is 28.7 minutes (46 seconds per cluster).

To understand the "consistency" of the descriptions, we let 3 external experts independently read and assess the collected descriptions. These experts are graduate students recruited outside of our research group (to avoid bias) and none of them participated in labeling clusters in the first round. For each cluster, an expert reads all 15 descriptions and assigns a consistency score (0 to 1), which is the ratio of the maximum number of consistent descriptions over all descriptions. For example, if 10 out of the 15 descriptions are consistent, the score is 10/15=0.667. The final consistency score is averaged over three experts. Figure 4.26 shows the consistency score distribution. The most common scores range from 0.6 to 0.8. The score distribution skews heavily to the right. This indicates that most clusters can be interpreted consistently.

Upon examining clusters with low consistency scores, we have two key observations. First, lower-level clusters are more difficult to interpret. As shown in Figure 4.27, average consistency scores decrease as we move further along the tree hierarchy. Intuitively, lower-level clusters represent more specific or even outlier behavior that is difficult to describe. Second, we find clusters with more selected features are harder to interpret. We perform correlation analysis between the consistency score and the number of selected features per cluster, and find they correlate negatively (Pearson coefficient r =-0.1, p =0.5). Noticeably, the consistency score also correlates negatively with the unique event types in selected features (Pearson coefficient r =-0.4, p =0.02).

Finally, we add short labels to the top-level clusters in Whisper and Renren based on the descriptions from user study and our own interpretations. The labels are shown in Figure 4.22 and Figure 4.23 respectively.

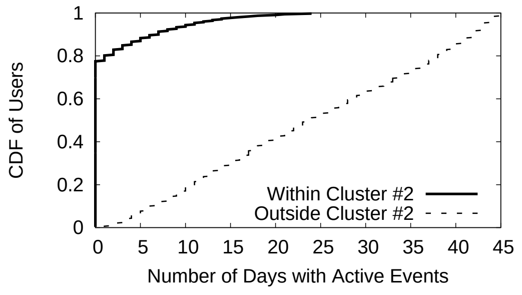


Figure 4.28: Number of days when users have active events in their clickstreams.

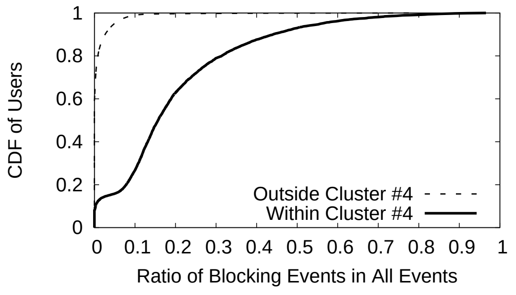


Figure 4.29: Ratio of blocking event over all click events in a user's clickstream.

4.2.5 Evaluation: Case Studies

Next, we present in-depth analysis on a few behavioral clusters as case studies. We have two goals. First, by analyzing the user behavior in these clusters, we validate the correctness of our cluster labels. Second, we explore the interesting (or unexpected) user behavior, and demonstrate the prediction power of the user behavioral models. Due to space limitation, we focus on two clusters from Whisper (Cluster#2 and Cluster#4), and one from Renren (Sybil Cluster).

4.2.5.1 Case Study 1: Inactive Whisper Users

We start with Cluster#2, which is labeled as inactive users. The selected action patterns of this cluster consist almost entirely of "receiving notification" events, indicating these users have not been actively engaged with the app. This is also confirmed in Figure 4.28: users in Cluster#2 have far fewer active days (when users actively generate clicks) than the rest of users. A remarkable 80% of users in Cluster#2 did not generate any active events through the 45 days, representing completely dormant users. In fact, our algorithm successfully groups dormant users into a separate subcluster (Figure 4.22, the biggest subcluster in Cluster#2).

Contrary to expectation, inactive users are not outliers. Cluster#2 is the second largest cluster with 21,962 users (20% of all users). From the perspective of service providers, it is important to identify the early signals of user disengagement, and implement mechanisms to re-gain user activities.

Predicting Dormant Users. We demonstrate the effectiveness of our behavioral models in predicting future user dormancy. The high-level idea is simple: Whisper can build behavioral models using users' most recent clickstreams, and update the models at regular intervals (*e.g.*, every month). Our hypothesis is that users placed in the "inactive" cluster are more likely to turn completely dormant. Thus we can use the inactive cluster to predict future dormant users.

We validate this hypothesis by investigating whether users in the "inactive" cluster will migrate to the "dormant" cluster over time. To do so, we split our clickstream data by date into three snapshots: Oct.13–27, Oct.28–Nov.12 and Nov.13–26. Then we generate behavioral clusters for each snapshot. The inactive cluster can be easily pinpointed within each snapshot based on selected activity patterns (*i.e.*, notification events). Also, we consistently find the following sub-structures within the inactive cluster: a big "dormant" cluster in which users have zero active events, alongside several "semi-dormant" clusters in which users are occasionally active.

In Table 4.6, we compare clusters from two adjacent snapshots to determine the likelihood of users migrating into the dormant cluster. The results confirm our hypothesis: Users in semi-dormant clusters are more likely to migrate to the dormant cluster than others. For example, 17% of semi-dormant users in snapshot-2 end up in the dormant cluster in snapshot-3, while only 1.1% of other users do so. Users already within the dormant cluster are highly likely to remain there through future snapshots (94%-99%). This result shows that our behavioral models can successfully track and predict the dormancy of Whisper users. It allows service providers to make timely interventions before losing user participation.

Cluster	# (%) of Users Join the Dormant Cluster	
	Snap 1 $\rightarrow$ Snap 2	Snap 2 $\rightarrow$ Snap 3
Dormant Cluster	15873/16872 (94%)	16161/16314 (99%)
Semi-dormant Clusters	363/9383 (4%)	2026/11773 (17%)
Other Clusters	63/73735 (0.09%)	804/71903 (1.1%)

Table 4.6: Users becoming dormant over time. We split the clickstream data into three snapshots, and report the number of users who migrate to the dormant cluster over two adjacent snapshots.

Actions per day	Statistics: Mean (STD)		T-statistics (p value)
	Inside C#4	Outside C#4	In vs. Out
Whisper Posted	1.25 (1.77)	0.65 (1.46)	27.43 ($p<0.001$)*
Replies Received	0.70 (4.09)	0.26 (1.41)	8.89 ($p<0.001$)*
Heart Received	2.39 (48.68)	0.69 (5.34)	2.93 ($p=0.0034$)*
Chats Initiated	2.20 (10.93)	1.18 (3.98)	7.79 ($p<0.001$)*

Table 4.7: Activity statistics for users inside and outside Cluster#4. *The difference is statistically significant based on Welch two-sample t-tests.

4.2.5.2 Case Study 2: Hostile Behaviors of Whisper Chatters

Next, we analyze Cluster#4, which contains 7026 users who tend to block other people during private chat. As shown in Figure 4.29, users in this cluster perform blocking actions much more frequently. 80% of users spend more than 10% of their total clicks on blocking events. In contrast, only 1% of users outside Cluster#4 achieve this ratio.

Next, we explore the possible causes to the blocking events. A private chat is initiated by the user who wants to talk to whisper authors. Our hypothesis is that users in Cluster#4 are more likely to post whispers which attract unwanted chatters to harass them. To validate this, we list behavioral statistics for users inside and outside Cluster#4 in Table 4.7. Users in Cluster#4 are more active in posting public whispers, which attract more hearts and replies from others (statistically significant based on Welch t-tests). These users are likely to experience harassment as a side effect.

Users may attract unwanted chat messages due to the topics they write about. We analyze users' whisper content in Cluster#4 and find they often consist of sexually explicit messages

Users	Top 30 Keywords
Inside C#4	20f, 19f, 18f, 17f, 29, f, roleplay, daddy, wet, role, lesbians, 17, lesbian, kinky, trade, bored, kik, weakness, nude, threesome, bestfriend, msg, shower, boys, chubby, nipples, horny, female, dirty, message
Outside C#4	religion, que, bullshit, 18m, personally, bible, eventually, faith, sign, plenty, hilarious, congratulations, gender, brain, idiot, dumbass, ignorant, quite, depends, animals, google, society, loss, count, health, sexuality, em, business, sound, foot

Table 4.8: Top whisper keywords for users in Cluster#4 and users outside Cluster#4.

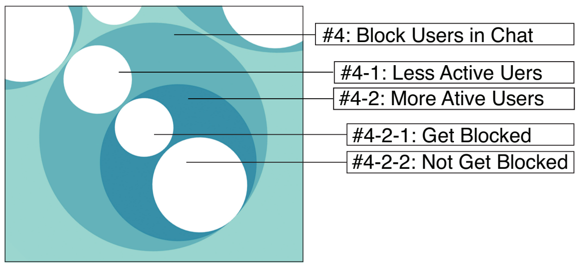


Figure 4.30: The sub-clusters within Cluster#4.

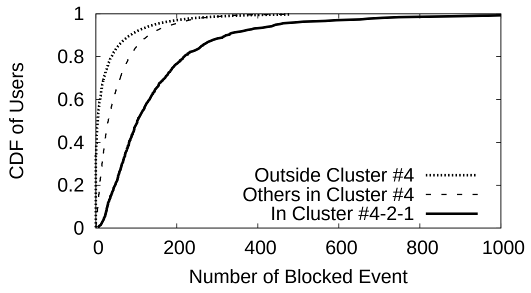
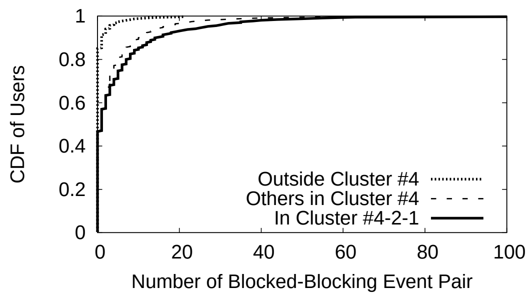


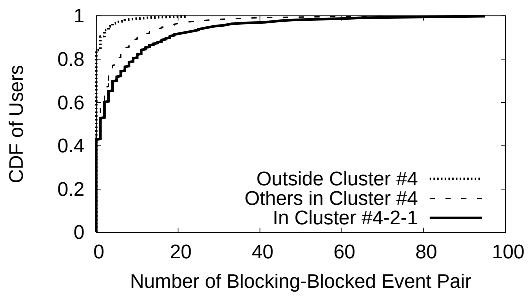
Figure 4.31: Number of being-blocked events per user.

(sexting). Table 4.8 lists top keywords from users in and outside Cluster#4. Keywords are ranked based on how strongly they are associated with the cluster. For each keyword, we compute a simple correlation ratio for ranking, as the number of whispers in Cluster#4 containing this keyword divided by the total number of whispers with this word. We exclude common stopwords [53] and low frequency words to avoid statistical outliers. A mere glance at Table 4.8 reveals that Cluster#4 users are focused on exchanging sexual content. Terms like “20f”, “f”,“17” and “lesbian” indicate age, gender (f = female) and sexual orientation. Other frequently used words are associated with the exchange of nude photos (“trade”, “shower”, “nipples”) or more general erotic terms.

Users Who Get Blocked. Within Cluster#4, we find a subcluster of 1412 users who often



(a) Blocked$\rightarrow$Blocking Event



(b) Blocking$\rightarrow$Blocked Event

Figure 4.32: Number paired blocking and blocked events per user. We match blocking and blocked events under the same whisper with time interval < 1 hour.

get blocked by others (Cluster#4-2-1 in Figure 4.30). As shown in Figure 4.31, these users have more "being-blocked" events in their clickstreams. In the meantime, as members of Cluster#4, these users are also highly likely to block other users.

Then the question is how often blocks are "bidirectional", *i.e.*, user X blocks Y and then Y immediately blocks X. Unfortunately, our dataset cannot directly measure bidirectional blocks. For a blocking event, the known information includes the whisperID where two users chat, the userID issuing the block, but not the userID being blocked. Thus we take an approximation approach to match potential "bidirectional" blocks (as upper bound). For each user, we group her blocking and being-blocked events under the same whisperID as a *pair* if their time interval is within a short time window (*e.g.*, one hour). This approximates immediate blocking back after getting a block. Figure 4.32 shows the matching result using time window as 1-hour. Users in Cluster#4, particularly in Cluster#4-2-1 have a higher number of paired blocking events. It is likely these users are easily offended or often offend other users during private chat, suggesting a strong hostile behavior. We also test 10 minutes and 1-day time window and have similar conclusion.

4.2.5.3 Case Study 3: Renren Sybil Accounts

Finally, we analyze the Sybil cluster in Renren (Cluster#2 in Figure 4.23). Our system groups Sybil accounts into one single cluster with high accuracy. 95% of true Sybils are clustered into the cluster and only 0.74% of normal users are misclassified. The selected features indicate Sybils are more likely to engage in sending friend requests. This makes sense because a Sybil must first befriend a user before accessing private information or spamming.

In addition, our system uncovers more fine-grained subclusters within the Sybil cluster, representing different attack strategies. Here we focus on the largest 5 (out of 8 subclusters), which encompass 99.36% of Sybil accounts. Table 4.9 shows their behavioral statistics. First, S_3 appears to describe "crawlers" who specialize in collecting user information and photos for sale on the black market [155]. Second, S_1, S_2 and S_4 all focus on "sending friend requests." Sybils in S_1 send requests in bulks via Renren's friend recommendation system, resulting in a high volume of friend requests per day (25.13). On the other hand, Sybils in S_4 tend to build social connections slowly (8.76 requests per day), possibly to avoid being detected. Finally, Sybils in S_5 are likely to *receive* friend requests. The ratio of incoming friend requests over outgoing ones is notably higher (0.286) than other Sybil clusters (< 0.05). One possible explanation is that these Sybils are controlled by a single attacker to befriend with each other to bootstrap their social connections.

4.2.6 Evaluation: Cluster Quality

Finally, we evaluate the quality of behavioral clusters produced by our system by examining how well they capture similar users. For this analysis, we compare our algorithm with existing clustering methods.

ID	Cluster Label	# of Users	FrdReq per Day	ProfileReq per Day	In/out FrdReq
S_1	Friending in bulks	4064	**25.13**	0.30	0.002
S_2	Friending quickly	1891	19.81	2.08	0.004
S_3	Crawl profiles	1348	11.41	**6.44**	0.050
S_4	Friending slowly	899	**8.76**	1.93	0.00004
S_5	Receive FrdReq	129	25.65	3.43	**0.286**
#1	Normal users	6141	1.65	2.80	1.06

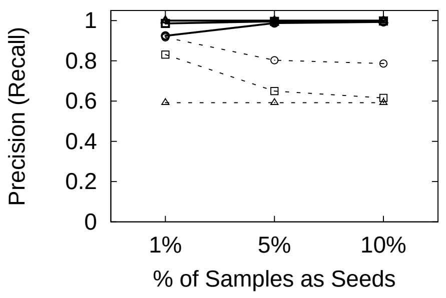
Table 4.9: Characteristics of users the 5 biggest Sybil clusters (S_1–S_5) and the normal user cluster. We add the cluster label based on the selected action patterns per cluster. "FrdReq" stands for "friend requests."

4.2.6.1 Clustering Quality

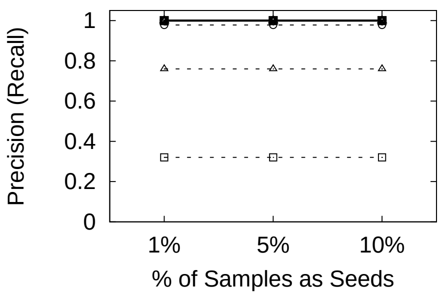
At the high-level, an effective clustering algorithm should accurately group similar users together while separating different ones. We evaluate the quality of our behavioral clusters by testing how well they capture similar users. More specifically, given a small sample of known users, how accurately can they retrieve other users of the same type?

Experiment Setups. We first explain our experiment method, using Sybil detection in Renren as an example. Suppose a small sample of Sybils are known to us (x%). To detect the rest of the Sybil accounts, we use the known samples as *seeds* to color Renren's behavioral clusters. Any cluster that contains a known Sybil will be colored as Sybil-cluster (uncolored ones as normal). We evaluate the accuracy using two metrics: *Precision* (percentage of users in Sybil-clusters that are true Sybil accounts) and *Recall* (percentage of true Sybils that are captured by Sybil-clusters). A higher precision and recall indicate a better clustering quality. We vary the parameter x (1%, 5%, 10%) and repeat each experiment 10 times.

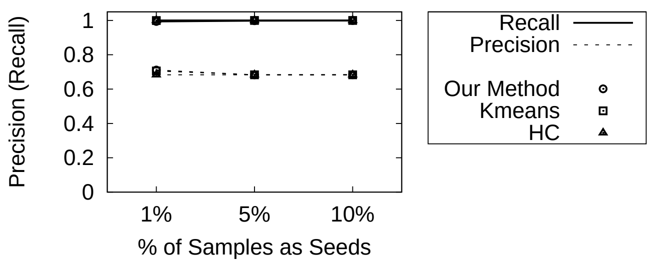
To perform this experiment on Whisper dataset, we need to construct known groups of users. We use the two types of users identified in earlier analysis: *Dormant* users who have zero active events (16688 users) and *Blocked* users who have been blocked at least once in a private chat (68302 users).



(a) Sybils (Renren)



(b) Dormant Users (Whisper)



(c) Blocked Users (Whisper)

Figure 4.33: The precision and recall of using the behavioral clusters to detect certain type of users. We compare our method with K-means and Hierarchical Clustering algorithm (HC).

Comparison Baselines. Our baselines are two widely used clustering algorithms: K-means[97] and Hierarchical Clustering (HC) [122]. We run both algorithms to cluster the full similarity graph (without feature pruning). At the high-level, K-means divides users into K clusters where each user is assigned to the nearest cluster (center). The number of clusters K must be pre-defined. Here we generate multiple versions of K-means clusters, and pick the K with the highest clustering quality (modularity). As a result, K-means generates 10 clusters on the Renren dataset and 10 for Whisper. In the same way, HC generates 7 clusters for Whisper and 2 clusters for Renren.

Results. First, for Sybil detection on Renren, our algorithm is highly accurate with a precision of 93% and a recall of 94% (1% ground-truth as seed) as shown in Figure 4.33a. Using more seeds (*e.g.* 5%) produces a higher recall (99%) but reduces precision (82%). Nonetheless, the overall performance is better than K-means and HC (precision 67% and 61%). On the Whisper dataset, our algorithm achieves accurate results (98% precision, 100% recall) in identifying dormant users (Figure 4.33b). K-means and HC have a much lower precision (32% and 78%) with the same recall. Finally, all three algorithms achieve similar accuracy in detecting blocked users (73% precision and 99% recall). These results indicate that our system produces high quality clusters to capture similar users.

4.2.7 Summary of Results

In this work, we describe a practical clickstream tool to model online user behavior. Our tool captures complex human behaviors while presenting them in a simple and intuitive manner. For a given clickstream dataset, it automatically identifies clusters of users with similar clickstream activities, and captures the natural hierarchical structure for user clusters. With a visualization tool, service providers can explore dominating user behaviors and categories as an overview, while tracking fine-grained user behavioral patterns along each category. Our tool does not require prior knowledge or assumptions of user categories (unsupervised), thus it can effectively capture unexpected or previously unknown behaviors. We demonstrate its effectiveness using case studies on two large-scale online social networks. Our tool accurately identifies unusual behaviors (malicious Sybils, hostile users) and even predicts users' future activities (dormant users). Finally, we shared our tool and results with the Whisper Data Science team. While we are awaiting more detailed comments, the initial feedback was extremely positive.

We believe our proposed techniques are generalizable beyond online social networks. To obtain clickstream traces, service providers can extract "user events" from their HTTP logs. In our analysis, we define user events based on social network features. For other services, specific events will depend on the service functionalities. For example, Wikipedia, News or Q&A sites might extract events based on the category or topic of the pages. E-commerce web sites can define user events based on the functionality of the clickable links or product categories. Crowdsourcing sites can define click events based on the crowdsourcing workflow. In future work, we will explore broader applications of clickstream behavioral models, and expand our tool to other user-driven systems.

Chapter 5

Related Work

5.1 Social Question Answering

Community based Q&A. Researchers have studied community based Q&A (CQA) sites such as Yahoo Answers [96, 35, 95, 150, 194, 195], MSN QnA [104, 183], Stack Overflow [39, 143], Math Overflow [212] from different perspectives. Some looks into managing questions and topics in CQA sites. Others study question archiving and tagging [183]. In addition, researchers have proposed methods to classify factual questions with conversional questions [95, 150], or reuse the knowledge collected from old questions to answer new similar questions [195]. Finally, others evaluate the quality of user generated content, including answer quality [194, 212, 35, 110] and question quality [39, 136].

Experts in Q&A Sites. A second group of work seeks to develop algorithms to identify experts in Q&A sites. One direction is to rank users based on expertise measures generated from user history data (*e.g.* questions, answers, votes) [35, 170, 137]. Another direction is modeling user interaction to design network-based ranking algorithms to identify experts [118, 134, 248]. Finally, other works study user community from perspectives such as answering speed [143] and user incentives in CQA sites [104].

Q&A in Social Networks. Studies have also looked into the question and answering behaviors in existing online social networks. Users can ask their friends questions by posting tweets in Twitter [173] or updating status in Facebook [171, 154, 102].

Our work (Section 2.1) is the first to analyze a social network based Q&A site using large-scale data measurement and analysis. Instead of treating all users as one big community, we explore the impact of a built-in social network and other graph structures. A recent report [174] looks at Quora's reputation system in depth with a small dataset of 5K questions.

5.2 Online Social Networks and Anonymity

Over the last few years, researchers have performed measurement studies on online social networks (OSNs) including Facebook [236, 218], Twitter [61, 128], Pinterest [82], and Tumblr [62]. Most of today's online social networks have stored large volumes of sensitive data about users (*e.g.*, personal profile, friending information, activity traces), all of which pose potential privacy risks. Various techniques have been proposed to compromise user anonymity and infer users' sensitive information from social network data [158, 152, 253, 43].

Anonymous online social networks such as Whisper and Yik Yak allow users to post content and communicate without revealing their real identity. Prior works have studied various anonymous platforms including anonymous forums [193], discussion boards [48, 124, 162] and Q&A sites [103]. Most earlier works study user communities focusing on content and sentiment analysis. More recently, anonymous social networks have emerged, particularly on mobile platforms. A recent work [184] conducted a user survey on SnapChat to understand how they used the anonymous social app. Another recent user study explores the correlation between content intimacy (real-name or anonymous) and willingness to self-disclose [141] In comparison, our study (Section 2.2) is the first to *quantitively* study user interaction, user engagement, and security implications in the anonymous Whisper network.

5.3 Security and Privacy in Location based Services

Attacks on Location based Services. Location-based services face various threats, ranging from rogue users reporting fake GPS [59, 98], to malicious parties compromising user privacy [70, 125, 126]. A related study on Waze [196] demonstrated that small-scale attacks can create traffic jams or track user icons, with up to 15 mobile emulators. Our work (Section 2.3) differs in two key aspects. First, we show that it's possible to reverse engineer its APIs, enabling light-weight Sybil devices (simple scripts) to replace full-stack emulators. This increase the scale of potential attacks by orders of magnitude, to thousands of Waze clients per commodity laptop. The impact of thousands of virtual vehicles is qualitatively different from 10-15 mobile simulators. Second, as possible defenses, [196] cites known tools such as phone number/IP verification, or location authentication with cellular towers, which have limited applicability.

Protecting Location Privacy against Service Providers. Researchers have proposed to preserve user location privacy against map services such as Waze and Google. Earlier studies apply location cloaking by adding noise to the GPS reports [90]. Recent work use zero-knowledge [111] and differential privacy [56] to preserve the location privacy of individual users, while maintaining user accountability and the accuracy of aggregated statistics. Our work differs by focusing on the attacks against the map services.

Mobile Location Authentication. Defending against forged GPS is challenging. One direction is to authenticate user locations using wireless infrastructures: WiFi APs [140, 191], cellular base stations [140, 191] and femtocells [54]. Devices must come into physical proximity to these infrastructures to be authenticated. But it requires cooperation among a wide range of infrastructures (also modifications to their software/hardware), which is impractical for large-scale services like Waze. Our work (Section 2.3) only uses a small number of trusted infrastructures to bootstrap, and relies on peer-based trust propagation to achieve coverage.

Other researchers have proposed “peer-based” methods to authenticate collocated mobile devices [211, 233, 254, 144, 159].

5.4 Spam, Malicious Crowdsourcing and Sybil Detection

OSN Spam and Detection. Researchers have identified copious amounts of fake accounts and spam campaigns on large OSNs like Facebook [77], Twitter [88, 213], and Renren [244]. The growing threat posed by this malicious activity has spurred work that aims to detect and stop OSN spam using machine learning techniques [46, 222, 206]. This body of research has focused on analyzing and defending against the outward manifestations of OSN spam. In contrast, our work (Section 3.1) identifies some of the underlying systems used by attackers to generate spam and evade security measures.

Opinion Spam. Spam that attempts to influence the opinions and actions of normal people has become more prevalent in recent years [113]. Researchers have been working on detecting and characterizing fake product reviews [138, 114], fake comments on news sites [64], and astroturf political campaigns on Twitter [180]. The authors of [169] created a model to help classify deceptive reviews generated by Mechanical Turk workers. These works reaffirm our results (Section 3.1), that crowdturfing is a growing, global threat on the web.

Crowdsourcing and Crowdturfing. Since coming online in 2005, Amazon’s Mechanical Turk has been scrutinized by the research community. This includes studies of worker demographics [106, 185], task pricing [76, 108], and even meta-studies on how to use Mechanical Turk to conduct user studies [123]. The characteristics of Micro Workers have also been thoroughly studied [101].

Our work is among the first to look into the misuse of crowdsourcing for malicious campaigns in online services. After our work, other researchers have conducted measurements on different crowdturfing sites to understand their operation and economic structure [132, 133, 157]. Some systems have been developed to detect paid human spammers in online review sites [169] and Q&A systems [63]. We are also among the first to explore detection of crowdturfing behaviors in adversarial settings (Section 3.2).

Sybil Detection. Sybils or fake accounts are the fundation of many online attacks (*e.g.*, spam, malware distribution) and have become a significant threat in online services [71]. In the context of online social networks, most Sybil detection systems rely on social graphs [247, 246, 217, 221, 69, 58]. These systems detect tight-knit Sybil communities that have a small quotient-cut from the honest region of the graph. 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 [244]. 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 [153].

A second body of work has used machine learning to detect Sybil behavior on Twitter [245, 46, 222] and Facebook [206]. However, relying on specific features makes these systems vulnerable to Sybils with different attack strategies. Finally, one of my earlier work proposes using crowdsourcing to identify Sybils [225]. In Section 4.1, our clickstream based Sybil detection is semi-unsupervised, which does not rely on specific assumptions about Sybil behaviors, and thus can detect previously unknown Sybils.

5.5 Adversarial Machine Learning

In an early study [105], 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 [50, 68, 142, 161] with less focusing on *causative* attacks [51, 186], since it's usually more difficult for adversaries to access training data in practice.

Several studies have examined attacks on specific ML-based applications, from email spam detection [68] to network intuition detection [186, 198, 220] to malicious (PDF) file classification [50, 142, 201, 240] and malware detection [120]. Our work (Section 3.2) focuses on crowdturfing and explores a wider range of adversarial attacks, including active evasion and more powerful poison attacks against the model training process.

5.6 Clickstream Analysis and User Behavior Study

Understanding Web Usage via Log Analysis. Understanding user behavior is important to the design and operation of online services. Recent works analyze network traces or logs to understand online users' browsing habits [166, 36]. Researchers also built more specific user behavioral models to study users' search intent [172] and Wikipedia editing patterns [80], to predict crowdsourcing worker performance [188].

Clickstream Analysis. Earlier research also used clickstream data for Web Usage Mining [200]. Researchers applied simple methods such as Markov Chains to capture users' navigation paths within a website [139, 189, 47]. However, these models focus on the simple aspects of user behavior (*e.g.*, user's favorite webpage), and are incapable of modeling more sophisticated user behavior. Other approaches use clustering techniques to identify user groups that share similar clickstream activities [216, 224, 209, 91]. The resulting clusters can be used to infer user interests [209] or predict future user behaviors [91]. However, existing clustering based models are largely supervised (or semi-supervised), requiring large samples of ground-truth data to train or fine-tune the model parameters [224, 216, 189]. Also, many behavioral models are built as "black boxes" for classification tasks, offering little explanations on how users behave and why [91]. Our work in Section 4.2 seeks to build unsupervised clickstream behavioral models and produce intuitive explanations on the models.

Clickstream Visualization. Researchers have developed interactive interfaces to visualize and inspect clickstream data. Existing tools generally focus on visualizing raw user clicks [147], click event sequences [252] or click transitions [235]. Instead, we build a tool (Section 4.2) to visualize the clickstream behavioral clusters produced by our system, providing hints for understanding key user behavior patterns.

Chapter 6

Conclusion

In this chapter, we summarize the key research results of this dissertation and discuss future directions. As exemplified in previous chapters, the main goals of my work are to identity the incorrect assumptions and vulnerabilities in existing online services by collecting and analyzing real-world data, and leverage theses insights to design and deploy new security solutions. In the following, I will first discuss data-driven security based on my own experiences with focus on its impact and open challenges. Then I will briefly discuss my future research plans along this path.

6.1 Data-driven Security.

Impact of Measurements. The biggest impact of data-driven research is to identify key mistakes or even failures in real-world systems which will ultimately drive the development of new and more effective systems. In this dissertation, we have demonstrated multiple successful examples. In Chapter 2, our measurements reveal how graph structures within Quora system helps to foster relevant and high-quality content (and why other services fail to do so); Our analysis of Whisper network shows the anonymity features do introduce more abusive content (*e.g.*, sexting), and the location fuzzing techniques are far from sufficient to protect user location privacy; Similarly, our measurement on Waze identities a scalable approach to create a large army of Sybil devices, which leads to new vulnerabilities for manipulating real-time traffic and massively tracking user movements. Practical solutions to this problem still remains to be found at this stage. A final example is Chapter 3 where we systematically quantify the organization and end-to-end impact of crowdturfing campaigns. The key takeaway is that human-based attackers are posing a significant threat to online communities, as most defenses systems (*e.g.*, CAPTCHA, rate-limit or template-based spam filters) assume attackers are automated software, and thus are vulnerable to malicious human users. All these successful examples confirm the need to use data-driven analysis as the basic tool in developing the next generation of security mechanisms on the Internent.

Novel Security Systems. Based on insights from large data analytics, we then develop new security systems to address the security and privacy issues in online communities. In Chapter 4, we use a large-scale ground truth dataset to develop machine learning models to detect malicious crowdsourcing workers. The best ML models can effectively detect regular workers (95% accuracy) or "professionals" (99% accuracy). More importantly, we use crowdturfing defense as context to explore the robustness of ML algorithms against adversarial attacks. We 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.

Finally, we develop clickstream user behavior models as a clickstream "similarity graph" which captures clusters of users with similar activities. On top of the model, we build a practical Sybil detector that requires minimal ground-truth data to bootstrap and have successfully identified previously unknown attacks in real-world social networks such as Renren and Linkedin. In addition, we extend the model to capture more fine-grained user behavior groups by constructing a hierarchical structure for clusters, and automatically extracting key features to interprete the meaning of captured clusters. Other than the successful applications in online social networks, our on-going works have produced successful results in other domains such as profiling investors in the stock market, enhancing wireless measurements with crowdsourcing efforts, and trace-driven user mobility models.

Challenges and Open Questions. Data-driven security analysis as a useful tool also has practical challenges. First, data collection. Although more and more Interent data become publicly available, certain data such as per-user browsing traces and internal service logs are still largely not available to researchers due to strong privacy implications. We are fortunate to get access to some of the non-public data via our collaborations with companies (*e.g.*, Whisper, Renren), which usually takes years to establish the trust for a workable relationship. Another related challenge is the research results based on non-public data is difficult to be repeated or reproduced by other researchers. This is a common problem for research papers published by companies such as Facebook and Google whose data is often not available for sharing within research communities.

A second challenge is to obtain ground-truth. Among all the available datasets such as social graphs, user interaction traces, most of them are not "labelled", especially those that are related to security events. Some researchers rely on public blacklists provided by companies such as Google to verify malicious URLs or websites, while others deploy "honeypot" to passively wait for attackers to hit their pre-set targets. In our own projects, we have also proactively interacted with attackers to collect ground-truth. For example, we have used crowdturfing services to run benign campaigns to understand end-to-end impact of online spam. Often cases, it is very difficult to assess whether the ground-truth data is sufficiently large and representative, whether it is biased or suffers from data noise. To theses ends, we usually need multiple, independently collected ground-truth datasets.

A third challenge is to assess the impact of errors. In security contexts (*e.g.*, attack detection), false positive and fale negative are commonly used to quantify the level of errors of a system. However, those numbers can have different practical meanings in different scenarios. For example, for Sybil detection systems, incorrectly blocking or banning a real, innocent user from the service (false postive) is usually a more serious error than missing a Sybil account (false negative). The real challenge is how to quantify the intuitive differences regarding the impact of the errors and incorporate that into the system design. This is not a easy task for either security system designers nor the end-users who are using the system.

A final challenge is the lack of longitudinal data to study the dynamic changes of attacker behaviors in online communities. For instance, many existing works (some of mine included) use datasets that are only in the length of weeks or months, which poses limits to the understanding of the problem. In the future, with an increasing attention on big data analytics and more advanced data mining tools, both research communities and industries are likely to collect more longitudinal data such as network traces, security incidents, etc. More research effort is needed for developing systematically tools to process and analyze streamed and longitudinal data.

6.2 Future Directions

Looking forward, I plan to continue to work on data-driven techniques to understand and address security issues in Internet systems. My plan is to extend large-scale data analytics to broader security contexts. Specific projects include ubiquitous crowdsourcing system design, and big data analytics for human centric security.

Ubiquitous crowdsourcing & security. Crowdsourcing has expanded beyond the Internet and become ubiquitous in the physical world, e.g., house cleaning (TaskRabbit), package delivery (Postmates), taxi services (Uber) and shared parking (Rover). Future crowdsourcing systems would involve both complex online user activities and offline interactions with ubiquitous physical facilities such as mobile devices, vehicles and buildings. To establish trust in the system and prevent malicious attacks, I plan to work on robust reputation systems that incorporate both online and offline data, and security mechanisms to continuously monitor the behavior of crowdsourcing participants and detect malicious actions in real-time.

Another direction of interest is to explore fundamental design choices for crowdsourcing systems to reduce malicious behavior. The initial step is to understand the tradeoffs of competition and cooperation among crowdsourcing participants. Existing systems (e.g., Amazon Turk) implement strong competitions by default, which however makes it extremely difficult for newcomers to survive and inevitably foster malicious behavior. Future research will explore novel designs to reduce malicious competition and improve the overall efficiency with structured cooperation.

Big data in human-centric security. In the long term, as computations are increasingly human-centric, future security systems will be having more direct and intimate interactions with users. A successful security system should be usable in the hand of large-scale online users. I believe big data analytics can make a difference in the design and deployment of usable security systems. On one hand, by collecting and analyzing large-scale data on security system usage, I seek a deep understanding on user-level misuse and misconfigurations of security systems, and explore key design flaws. On the other hand, I plan to collaborate with system and HCI researchers to build novel data-driven security mechanisms that offer high-level transparency and usability. This includes intelligent underlying algorithms to identify security threats from continuous data analytics, and advanced data visualization techniques to help users fully understand the emerging threats to make informed security decisions.

Bibliography

[1] "Charles Proxy." http://www.charlesproxy.com.

[2] "GenyMotion Emulator." http://www.genymotion.com.

[3] "Monkeyrunner." http://developer.android.com/tools/help/monkeyrunner_concepts.html.

[4] "Open Source Routing Machine (OSRM)." http://map.project-osrm.org.

[5] "Waze's response to our research." https://blog.waze.com/2016/04/privacy-and-waze.html.

[6] "Zeus botnet targets facebook." Appriver.com, 2009. http://blog.appriver.com/2009/10/zeus-botnet-targets-facebook/.

[7] "China's internet users targeted in online rumour probes." BBC, 2011.

[8] "Websites shut over illegal PR deals." China Daily, 2011.

[9] "What is quora algorithm/formula for determining the ordering or ranking of answers on a question?." Quora, Feb., 2011. http://www.quora.com/Quora-product/What-is-Quoras-algorithm-formula-for-determining-the-ordering-ranking-of-answers-on-a-question.

[10] "10 people you wont believe have fake followers on twitter." AaBaco Small Business, 2012. https://www.aabacosmallbusiness.com/advisor/blogs/profit-minded/10-people-won-t-believe-fake-followers-twitter-215539518.html.

[11] "Bullying on twitter: Researchers find 15,000 bully-related tweets sent daily." Huffington Post, 2012. http://www.huffingtonpost.com/2012/08/02/bullying-on-twitter_n_1732952.html.

[12] "Facebook has more than 83 million illegitimate accounts." BBC News, August, 2012.

[13] "How many questions are on quora, answered or not?." Quora, Mar., 2012. http://www.quora.com/Quora-Usage-Data-and-Analysis/How-many-questions-are-on-Quora-answered-or-not.

[14] "Mitt romney suspiciously gets 116k twitter followers in one day." CNet, 2012. http://www.cnet.com/news/mitt-romney-suspiciously-gets-116k-twitter-followers-in-one-day/.

[15] "Obama has millions of fake twitter followers." USA Today, 2012. http://content.usatoday.com/communities/theoval/post/2012/08/obama-has-millions-of-fake-twitter-followers/1.

[16] "On what topics does quora have the best questions and answers?." Quora, Apr., 2012. http://www.quora.com/Lists-of-Top-Quora-Content/On-what-topics-does-Quora-have-the-best-questions-and-answers.

[17] "What is a closed question?." Stack Overflow, Oct., 2012. http://meta.stackoverflow.com/questions/10582/what-is-a-closed-question.

[18] "What is quora's policy on question deletion?." Quora, Jun., 2012. http://www.quora.com/Quora-Policies-and-Guidelines/What-is-Quoras-policy-on-question-deletion.

[19] "Who are all of the reviewers on quora?." Quora, Nov., 2012. http://www.quora.com/Quora-Reviewers/Who-are-all-of-the-reviewers-on-Quora.

[20] "Who are the current site admins of quora?." Quora, Aug., 2012. http://www.quora.com/Quora-Admins/Who-are-the-current-site-admins-of-Quora.

[21] "Sina Weibo Terms of Service." http://service.account.weibo.com/roles/guiding, 2013. (the link is accessible after login).

[22] "Verify facebook account." https://www.facebook.com/help/398085743567023/, 2013.

[23] "Anonymous mesaging app yik yak contributes to cyberbullying problem." Education News, 2014. http://www.educationnews.org/technology/anonymous-mesaging-app-yik-yak-contributes-to-cyberbullying-problem/.

[24] "A gossip app brought my high school to a halt." NY Mag, 2014. http://nymag.com/thecut/2014/04/gossip-app-brought-my-high-school-to-a-halt.html.

[25] "New home of cyberbullying? yik yak gossip app takes off in high schools." Today.com, 2014. http://www.today.com/parents/new-home-cyberbullying-yik-yak-gossip-app-takes-high-schools-2D79597218.

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

[27] "Facebooks war continues against fake profiles and bots." Appriver.com, 2015. http://www.huffingtonpost.com/james-parsons/facebooks-war-continues-against-fake-profiles-and-bots_b_6914282.html.

[28] "Social media gives clues to security questions." USA Today, 2015. http://www.usatoday.com/story/money/2015/06/01/irs-breach-personal-data-vulnerable/28068875/.

[29] "55 companies and counting – w2 spear phishing attacks continue to increase." Cloudmark.com, 2016. https://blog.cloudmark.com/2016/03/31/55-companies-and-counting-w-2-spear-phishing-attacks-continue-to-increase/.

[30] "Facebook statistics." http://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/, 2016.

[31] "Ftc announces significant enhancements to identitytheft.gov." FTC, 2016. https://www.ftc.gov/news-events/press-releases/2016/01/ftc-announces-significant-enhancements-identitytheftgov.

[32] "Main line health employees targeted in 'spear phishing' scheme." NBC Philadelphia, 2016. http://www.nbcphiladelphia.com/news/local/Spear-Phishing-Main-Line-Health-Employees-Pennsylvania-Philadelphia--370867511.html.

[33] "Snapchat's video traffic is catching facebook." Fortune, 2016. http://fortune.com/2016/01/12/snapchat-facebook-video-views/.

[34] "Twitter statistics." http://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users/, 2016.

[35] L. A. Adamic, J. Zhang, E. Bakshy, and M. S. Ackerman, *Knowledge sharing and yahoo answers: everyone knows something.*, in *Proc. of WWW*, 2008.

[36] E. Adar, J. Teevan, and S. T. Dumais, *Large scale analysis of web revisitation patterns*, in *Proc. of CHI*, 2008.

[37] Y.-Y. Ahn, S. Han, H. Kwak, S. Moon, and H. Jeong, *Analysis of topological characteristics of huge online social networking services*, in *Proc. of WWW*, 2007.

[38] H. Almuhimedi, S. Wilson, B. Liu, N. Sadeh, and A. Acquisti, *Tweets are forever: a large-scale quantitative analysis of deleted tweets*, in *Proc. of CSCW*, 2013.

[39] A. Anderson, D. Huttenlocher, J. Kleinberg, and J. Leskovec, *Discovering value from community activity on focused question answering sites: a case study of stack overflow*, in *Proc. of KDD*, 2012.

[40] M. Andreesen, "Public tweets." Twitter, March, 2014. `https://twitter.com/pmarca/status/444941023710433280`.

[41] D. Ashbrook and T. Starner, *Using gps to learn significant locations and predict movement across multiple users*, *Personal Ubiquitous Comput.* **7** (2003), no. 5 275–286.

[42] Associated Press, "Whispers, secrets and lies? anonymity apps rise." USA Today, March, 2014. `http://www.usatoday.com/story/tech/personal/2014/03/25/anonymity-apps-rise/6863433`.

[43] L. Backstrom, C. Dwork, and J. Kleinberg, *Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography*, in *Proc. of WWW*, 2007.

[44] A. Banerjee and J. Ghosh, *Concept-based clustering of clickstream data*, in *Proc. of ICIT*, 2000.

[45] A.-L. Barabasi and R. Albert, *Emergence of scaling in random networks*, *Science* **286** (1999).

[46] F. Benevenuto, G. Magno, T. Rodrigues, and V. Almeida, *Detecting spammers on twitter*, in *Proc. of CEAS*, 2010.

[47] F. Benevenuto, T. Rodrigues, M. Cha, and V. Almeida, *Characterizing user behavior in online social networks*, in *Proc. of IMC*, 2009.

[48] M. S. Bernstein, A. Monroy-Hernández, D. Harry, P. André, K. Panovich, and G. G. Vargas, *4chan and/b: An analysis of anonymity and ephemerality in a large online community*, in *Proc. of ICWSM*, 2011.

[49] N. Bhatti, A. Bouch, and A. Kuchinsky, *Integrating user-perceived quality into web server design*, *Computer Networks* **33** (2000), no. 16 1 – 16.

[50] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Srndic, P. Laskov, G. Giacinto, and F. Roli, *Evasion attacks against machine learning at test time*, in *Proc. of ECML PKDD*, pp. 387–402, 2013.

[51] B. Biggio, B. Nelson, and P. Laskov, *Poisoning attacks against support vector machines*, in *Proc. of ICML*, 2012.

[52] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, *Fast unfolding of communities in large networks*, *JSTAT* **2008** (2008), no. 10.

[53] A. Brahaj, "English stop words." `http://xpo6.com/list-of-english-stop-words/`, 2009.

[54] J. Brassil, P. K. Manadhata, and R. Netravali, *Traffic signature-based mobile device location authentication*, *IEEE Transactions on Mobile Computing* **13** (2014), no. 9 2156–2169.

[55] L. Breiman, *Random forests*, *Machine Learning* **45** (2001), no. 1 5–32.

[56] J. W. S. Brown, O. Ohrimenko, and R. Tamassia, *Haze: Privacy-preserving real-time traffic statistics*, in *Proc. of SIGSPATIAL*, 2013.

[57] C. Brubaker, S. Jana, B. Ray, S. Khurshid, and V. Shmatikov, *Using frankencerts for automated adversarial testing of certificate validation in ssl/tls implementations*, in *Proc. of IEEE S&P*, 2014.

[58] Q. Cao, M. Sirivianos, X. Yang, and T. Pregueiro, *Aiding the detection of fake accounts in large scale social online services*, in *Proc. of NSDI*, 2012.

[59] B. Carbunar and R. Potharaju, *You unlocked the mt. everest badge on foursquare! countering location fraud in geosocial networks*, in *Proc. of MASS*, 2012.

[60] R. Caruana, N. Karampatziakis, and A. Yessenalina, *An empirical evaluation of supervised learning in high dimensions*, in *Proc. of ICML*, 2008.

[61] M. Cha, H. Haddadi, F. Benvenuto, and K. Gummadi, *Measuring User Influence in Twitter: The Million Follower Fallacy*, in *Proc. of ICWSM*, 2010.

[62] Y. Chang, L. Tang, Y. Inagaki, and Y. Liu, *What is tumblr: A statistical overview and comparison*, *CoRR* **abs/1403.5206** (2014).

[63] C. Chen, K. Wu, V. Srinivasan, and K. B. R, *The best answers? think twice: Online detection of commercial campaigns in the cqa forums*, *CoRR* (2012).

[64] C. Chen, K. Wu, V. Srinivasan, and X. Zhang, *Battling the internet water army: Detection of hidden paid posters*, *CoRR* **abs/1111.4297** (2011).

[65] A. Clauset, C. R. Shalizi, and M. E. Newman, *Power-law distributions in empirical data*, *SIAM review* **51** (2009), no. 4 661–703.

[66] D. Correa, L. A. Silva, M. Mondal, F. Benevenuto, and K. P. Gummadi, *The many shades of anonymity: Characterizing anonymous social media content.*, in *Proc. of ICWSM*, 2015.

[67] C. Dagum, *The Generation and Distribution of Income, the Lorenz Curve and the Gini Ratio*, *Economie Applique* **33** (1980) 327–367.

[68] N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma, *Adversarial classification*, in *Proc. of KDD*, 2004.

[69] G. Danezis and P. Mittal, *Sybilinfer: Detecting sybil nodes using social networks*, in *Proc of NDSS*, 2009.

[70] Y.-A. de Montjoye, M. Verleysen, and V. D. Blondel, *Unique in the crowd: The privacy bounds of human mobility*, *Scientific Reports* **3** (2013).

[71] J. R. Douceur, *The Sybil attack*, in *Proc. of IPTPS*, (Cambridge, MA), March, 2002.

[72] Y. Duan, "The invisible hands behind web postings." China Daily, 2010.

[73] K. Eaton, "Mechanical turk's unsavory side effect: Massive spam generation." Fast Company, 2010.

[74] W. Enck, D. Octeau, P. McDaniel, and S. Chaudhuri, *A study of android application security*, in *Proc. of USENIX Security*, 2011.

[75] S. Fahl, M. Harbach, H. Perl, M. Koetter, and M. Smith, *Rethinking ssl development in an appified world*, in *Proc. of CCS*, 2013.

[76] S. Faridani, B. Hartmann, and P. G. Ipeirotis, *What's the right price? pricing tasks for finishing on time*, in *Proc. of AAAI Workshop on Human Computation*, 2011.

[77] H. Gao, J. Hu, C. Wilson, Z. Li, Y. Chen, and B. Y. Zhao, *Detecting and characterizing social spam campaigns*, in *Proc. of IMC*, 2010.

[78] D. Garcia, P. Mavrodiev, and F. Schweitzer, *Social resilience in online communities: The autopsy of friendster*, in *Proc. of COSN*, 2013.

[79] L. Ge and T. Liu, "'water army' whistleblower threatened." Global Times, 2011.

[80] R. S. Geiger and A. Halfaker, *Using edit sessions to measure participation in wikipedia*, in *Proc. of CSCW*, 2013.

[81] S. Gianvecchio and H. Wang, *Detecting covert timing channels: an entropy-based approach*, in *Proc. of CCS*, 2007.

[82] E. Gilbert, S. Bakhshi, S. Chang, and L. Terveen, *"i need to try this!": A statistical overview of pinterest*, in *Proc. of CHI*, 2013.

[83] E. Gilbert and K. Karahalios, *Predicting tie strength with social media*, in *Proc. of CHI*, 2009.

[84] J. Giles, "Inside facebook's massive cyber-security system." New Scientist, 2011.

[85] V. Goel, "Maps that live and breathe with data." The New York Times, June, 2013.

[86] N. Z. Gong, W. Xu, L. Huang, P. Mittal, E. Stefanov, V. Sekar, and D. Song, *Evolution of social-attribute networks: measurements, modeling, and implications using google+*, in *Proc. of IMC*, 2012.

[87] Google, "Google maps and waze, outsmarting traffic together." Google Official Blog, June, 2013.

[88] C. Grier, K. Thomas, V. Paxson, and M. Zhang, *@spam: the underground on 140 characters or less*, in *Proc. of CCS*, 2010.

[89] J. V. Grove, "Secrets and lies: Whisper and the return of the anonymous app." CNet News, January, 2014.

[90] M. Gruteser and D. Grunwald, *Anonymous usage of location-based services through spatial and temporal cloaking*, in *Proc. of MobiSys*, 2003.

[91] S. Gündüz and M. T. Özsu, *A web page prediction model based on click-stream tree representation of user behavior*, in *Proc. of KDD*, 2003.

[92] L. Guo, E. Tan, S. Chen, X. Zhang, and Y. E. Zhao, *Analyzing patterns of user content generation in online social networks*, in *Proc. of KDD*, 2009.

[93] I. Guyon and A. Elisseeff, *An introduction to variable and feature selection*, *JMLR* **3** (2003) 1157–1182.

[94] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, *The weka data mining software: an update*, *SIGKDD Explor. Newsl.* **11** (2009), no. 1.

[95] F. M. Harper, D. Moy, and J. A. Konstan, *Facts or friends?: distinguishing informational and conversational questions in social Q&A sites*, in *Proc. of CHI*, 2009.

[96] F. M. Harper, D. Raban, S. Rafaeli, and J. A. Konstan, *Predictors of answer quality in online Q&A sites*, in *Proc. of CHI*, 2008.

[97] J. A. Hartigan and M. A. Wong, *Algorithm as 136: A k-means clustering algorithm*, *Applied statistics* (1979) 100–108.

[98] W. He, X. Liu, and M. Ren, *Location cheating: A security challenge to location-based social network services*, in *Proc. of ICDCS*, 2011.

[99] D. Heckerman, D. Geiger, and D. M. Chickering, *Learning bayesian networks: The combination of knowledge and statistical data*, *Mach. Learn.* **20** (1995), no. 3 197–243.

[100] J. Heer and E. H. Chi, *Separating the swarm: categorization methods for user sessions on the web*, in *Proc. of CHI*, 2002.

[101] M. Hirth, T. Hossfeld, and P. Tran-Gia, *Anatomy of a crowdsourcing platform - using the example of microworkers.com*, in *Proc. of IMIS*, 2011.

[102] D. Horowitz and S. D. Kamvar, *The anatomy of a large-scale social search engine*, in *Proc. of WWW*, 2010.

[103] H. Hosseinmardi, R. Han, Q. Lv, S. Mishra, and A. Ghasemianlangroodi, *Analyzing negative user behavior in a semi-anonymous social network*, *CoRR* **abs/1404.3839** (2014).

[104] G. Hsieh and S. Counts, *mimir: A market-based real-time question and answer service*, in *Proc. of CHI*, 2009.

[105] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. D. Tygar, *Adversarial machine learning*, in *Proc. of AISec*, 2011.

[106] P. G. Ipeirotis, “Demographics of mechanical turk.” NYU Working Paper, 2010.

[107] P. G. Ipeirotis, “Mechanical turk: Now with 40.92% spam.” Behind Enemy Lines blog, 2010.

[108] P. G. Ipeirotis, *Analyzing the amazon mechanical turk marketplace*, *XRDS* **17** (December, 2010) 16–21.

[109] D. Irani, M. Balduzzi, D. Balzarotti, E. Kirda, and C. Pu, *Reverse social engineering attacks in online social networks*, in *Proc of DIMVA*, 2011.

[110] J. Jeon, W. B. Croft, J. H. Lee, and S. Park, *A framework to predict the quality of answers with non-textual features*, in *Proc. of SIGIR*, 2006.

[111] T. Jeske, *Floating car data from smartphones: What google and waze know about you and how hackers can control traffic*, *Black Hat* (2013).

[112] J. Jiang, C. Wilson, X. Wang, P. Huang, W. Sha, Y. Dai, and B. Y. Zhao, *Understanding latent interactions in online social networks*, in *Proc. of IMC*, 2010.

[113] N. Jindal and B. Li, *Opinion spam and analysis*, in *Proc. of WSDM*, 2008.

[114] N. Jindal, B. Liu, and E.-P. Lim, *Finding unusual review patterns using unexpected rules*, in *Proc. of CIKM*, 2010.

[115] G. H. John and P. Langley, *Estimating continuous distributions in bayesian classifiers*, in *Proc. of UAI*, 1995.

[116] B. Johnson and B. Shneiderman, *Tree-maps: A space-filling approach to the visualization of hierarchical information structures*, in *Proc. of VIS*, 1991.

[117] J. J. Jones, J. E. Settle, R. M. Bond, C. J. Fariss, C. Marlow, and J. H. Fowler, *Inferring tie strength from online directed behavior*, *PLoS ONE* **8** (2013), no. 1 e52168.

[118] P. Jurczyk and E. Agichtein, *Discovering authorities in question answer communities by using link analysis*, in *Proc. of CIKM*, 2007.

[119] V. Kachitvichyanukul and B. W. Schmeiser, *Binomial random variate generation*, *Commun. ACM* **31** (1988), no. 2 216–222.

[120] A. Kantchelian, S. Afroz, L. Huang, A. C. Islam, B. Miller, M. C. Tschantz, R. Greenstadt, A. D. Joseph, and J. D. Tygar, *Approaches to adversarial drift*, in *Proc. of AISec*, 2013.

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

[122] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*, vol. 344. John Wiley & Sons, 2009.

[123] A. Kittur, H. Chi, and B. Suh, *Crowdsourcing user studies with mechanical turk*, in *Proc. of CHI*, 2008.

[124] L. Knuttila, *User unknown: 4chan, anonymity and contingency*, *First Monday* **16** (2011), no. 10.

[125] J. Krumm, *Inference attacks on location tracks*, in *Pervasive Computing*, pp. 127–143. 2007.

[126] J. Krumm, *A survey of computational location privacy*, *Personal and Ubiquitous Computing* (2009).

[127] J. B. Kruskal and J. M. Landwehr, *Icicle plots: Better displays for hierarchical clustering*, *The American Statistician* **37** (1983), no. 2 162–168.

[128] H. Kwak, C. Lee, H. Park, and S. Moon, *What is Twitter, a social network or a news media?*, in *Proc. of WWW*, 2010.

[129] A. Lakhina, M. Crovella, and C. Diot, *Diagnosing network-wide traffic anomalies*, in *Proc. of SIGCOMM*, 2004.

[130] A. Lau, “Reputation management: PR vs. search vs. china’s water army.” Search Engine Watch, 2011.

[131] Z. Le, “Average salaries up 13-14% last year as income disparity increases.” Global Times, 2011.

[132] K. Lee, P. Tamilarasan, and J. Caverlee, *Crowdturfers, campaigns, and social media: Tracking and revealing crowdsourced manipulation of social media,* in *Proc. of ICWSM,* 2013.

[133] K. Lee, S. Webb, and H. Ge, *The dark side of micro-task marketplaces: Characterizing fiverr and automatically detecting crowdturfing*, in *Proc. of ICWSM*, 2014.

[134] K. Lerman and A. Galstyan, *Analysis of social voting patterns on digg*, in *Proc. of WOSN*, 2008.

[135] M. Levandowsky and D. Winter, *Distance between sets*, *Nature* **234** (1971) 34–35.

[136] B. Li, T. Jin, M. R. Lyu, I. King, and B. Mak, *Analyzing and predicting question quality in community question answering services*, in *Proc. of CQA Workshop (WWW)*, 2012.

[137] B. Li and I. King, *Routing questions to appropriate answerers in community question answering services*, in *Proc. of CIKM*, 2010.

[138] E.-P. Lim, V.-A. Nguyen, N. Jindal, B. Liu, and H. W. Lauw, *Detecting product review spammers using rating behaviors*, in *Proc. of CIKM*, 2010.

[139] L. Lu, M. Dunham, and Y. Meng, *Mining significant usage patterns from clickstream data*, in *Proc. of WebKDD*, 2005.

[140] W. Luo and U. Hengartner, *Proving your location without giving up your privacy*, in *Proc. of HotMobile*, 2010.

[141] X. Ma, J. Hancock, and M. Naaman, *Anonymity, intimacy and self-disclosure in social media*, in *Proc. of CHI*, 2016.

[142] D. Maiorca, I. Corona, and G. Giacinto, *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.

[143] L. Mamykina, B. Manoim, M. Mittal, G. Hripcsak, and B. Hartmann, *Design lessons from the fastest Q&A site in the west*, in *Proc. of CHI*, 2011.

[144] J. Manweiler, R. Scudellari, and L. P. Cox, *Smile: Encounter-based trust for mobile social services*, in *Proc. of CCS*, 2009.

[145] C. Marforio, N. Karapanos, C. Soriente, and K. Capkun, *Smartphones as practical and secure location verification tokens for payments*, in *Proc. of NDSS*, 2014.

[146] M. Masnick, “Bot-on-bot ebay scamming.” Techdirt, 2006.

[147] J. Matejka, T. Grossman, and G. Fitzmaurice, *Patina: Dynamic heatmaps for visualizing application usage*, in *Proc. of CHI*, 2013.

[148] M. McGee, “Quora traffic has grown 882% in the last year, but its still just a fraction of what yahoo answers gets.” MarketingLand, Dec., 2011.

[149] M. McGee, “Yahoo answers hits 300 million questions, but Q&A activity is declining.” Search Engine Land, Jul., 2012.

[150] E. Mendes Rodrigues and N. Milic-Frayling, *Socializing or knowledge sharing?: characterizing social intent in community question answering*, in *Proc. of CIKM*, 2009.

[151] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharje, *Measurement and analysis of online social networks*, in *Proc. of IMC*, 2007.

[152] A. Mislove, B. Viswanath, K. P. Gummadi, and P. Druschel, *You are who you know: inferring user profiles in online social networks*, in *Proc. of WSDM*, 2010.

[153] A. Mohaisen, A. Yun, and Y. Kim, *Measuring the Mixing Time of Social Graphs*, in *Proc. of IMC*, 2010.

[154] M. R. Morris, J. Teevan, and K. Panovich, *What do people ask their social networks, and why?: a survey study of status message Q&A behavior*, in *Proc. of CHI*, 2010.

[155] M. Motoyama, D. McCoy, K. Levchenko, S. Savage, and G. M. Voelker, *An analysis of underground forums*, in *Proc. of IMC*, 2011.

[156] M. Motoyama, K. Levchenko, C. Kanich, D. McCoy, G. M. Voelker, and S. Savage, *Re: Captchas – understanding captcha-solving from an economic context*, in *Proc. of USENIX Security*, 2010.

[157] M. Motoyama, D. McCoy, K. Levchenko, S. Savage, and G. M. Voelker, *Dirty jobs: The role of freelance labor in web service abuse*, in *Proc. of Usenix Security*, 2011.

[158] A. Narayanan and V. Shmatikov, *Robust de-anonymization of large sparse datasets*, in *Proc. of IEEE S&P*, 2008.

[159] A. Narayanan, N. Thiagarajan, M. Lakhani, M. Hamburg, and D. Boneh, *Location Privacy via Private Proximity Testing*, in *Proc. of NDSS*, 2011.

[160] B. Nelson, B. I. P. Rubinstein, L. Huang, A. D. Joseph, S. hon Lau, S. J. Lee, S. Rao, A. Tran, and J. D. Tygar, *Near-optimal evasion of convex-inducing classifiers*, *CoRR* (2010).

[161] B. Nelson, B. I. P. Rubinstein, L. Huang, A. D. Joseph, S. J. Lee, S. Rao, and J. D. Tygar, *Query strategies for evading convex-inducing classifiers*, *J. Mach. Learn. Res.* **13** (2012), no. 1.

[162] E. Newell, D. Jurgens, H. Saleem, H. Vala, J. Sassine, C. Armstrong, and D. Ruths, *User migration in online social networks: A case study on reddit during a period of community unrest*, in *Proc. of ICWSM*, 2016.

[163] M. E. J. Newman, *Assortative mixing in networks*, *Physical Review Letters* **89** (2002), no. 20 208701.

[164] J. Newsome, *Polygraph: Automatically generating signatures for polymorphic worms*, in *Proc. of IEEE S&P*, 2005.

[165] V. Ni, "China's internet users hit 485 million, weibo users and group buyers surge." China Briefing, 2011.

[166] H. Obendorf, H. Weinreich, E. Herder, and M. Mayer, *Web page revisitation revisited: implications of a long-term click-stream study of browser usage*, in *Proc. of CHI*, 2007.

[167] J. Ong, "Chinas sina weibo grew 73% in 2012, passing 500 million registered accounts.." The Next Web, Feb., 2013.

[168] J. Osborne and A. Diquet, *When security gets in the way: Pentesting mobile apps that use certificate pinning*, *Black Hat* (2012).

[169] M. Ott, Y. Choi, C. Cardie, and J. T. Hancock, *Finding deceptive opinion spam by any stretch of the imagination*, in *Proc. of ACL*, 2011.

[170] A. Pal, S. Chang, and J. A. Konstan, *Evolution of experts in question answering communities*, in *Proc. of ICWSM*, 2012.

[171] K. Panovich, R. Miller, and D. Karger, *Tie strength in question & answer on social network sites*, in *Proc. of CSCW*, 2012.

[172] J. Y. Park, N. O'Hare, R. Schifanella, A. Jaimes, and C.-W. Chung, *A large-scale study of user image search behavior on the web*, in *Proc. of CHI*, 2015.

[173] S. A. Paul, L. Hong, and E. H. Chi, *Is twitter a good place for asking questions? a characterization study.*, in *Proc. of ICWSM*, 2011.

[174] S. A. Paul, L. Hong, and E. H. Chi, *Who is authoritative? understanding reputation mechanisms in quora*, in *Proc. of Collective Intelligence*, 2012.

[175] S. Petrovic, M. Osborne, and V. Lavrenko, *I wish i didn't say that! analyzing and predicting deleted messages in twitter*, *CoRR* **abs/1305.3107** (2013).

[176] J. C. Platt, *Fast training of support vector machines using sequential minimal optimization*, in *Advances in Kernel Methods - Support Vector Learning*, 1998.

[177] J. C. Platt, *Advances in kernel methods*, pp. 185–208. MIT Press, 1999.

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

[179] A. Ramachandran, N. Feamster, and S. Vempala, *Filtering spam with behavioral blacklisting*, in *Proc. of CCS*, 2007.

[180] J. Ratkiewicz, M. Conover, M. Meiss, B. Gonçalves, A. Flammini, and F. Menczer, *Detecting and tracking political abuse in social media*, in *Proc. of ICWSM*, 2011.

[181] B. Reed, "Google maps becomes googles second 1 billion-download hit." Yahoo! News, June, 2014.

[182] G. Robinson, *A statistical approach to the spam problem, Linux J.* **2003** (2003), no. 107.

[183] E. M. Rodrigues, N. Milic-Frayling, and B. Fortuna, *Social tagging behaviour in community-driven question answering*, in *Proc. of WI-IAT*, 2008.

[184] F. Roesner, B. T. Gill, and T. Kohno, *Sex, lies, or kittens? investigating the use of snapchats self-destructing messages*, in *Proc. of FC*, 2014.

[185] J. Ross, I. Irani, M. S. Silberman, A. Zaldivar, and B. Tomlinson, *Who are the crowdworkers?: Shifting demographics in amazon mechanical turk*, in *Proc. of CHI*, 2010.

[186] B. I. Rubinstein, B. Nelson, L. Huang, A. D. Joseph, S.-h. Lau, S. Rao, N. Taft, and J. D. Tygar, *Antidote: understanding and defending against poisoning of anomaly detectors*, in *Proc. of IMC*, 2009.

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

[188] J. M. Rzeszotarski and A. Kittur, *Instrumenting the crowd: Using implicit behavioral measures to predict task performance*, in *Proc. of UIST*, 2011.

[189] N. Sadagopan and J. Li, *Characterizing typical and atypical user sessions in clickstreams*, in *Proc. of WWW*, 2008.

[190] S. Salvador and P. Chan, *Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms*, in *Proc. of ICTAI*, 2004.

[191] S. Saroiu and A. Wolman, *Enabling new mobile applications with location proofs*, in *Proc. of HotMobile*, 2009.

[192] S. Saroiu and A. Wolman, *I am a sensor, and i approve this message*, in *Proc. of HotMobile*, 2010.

[193] S. Y. Schoenebeck, *The secret life of online moms: Anonymity and disinhibition on youbemom.com*, in *Proc. of ICWSM*, 2013.

[194] C. Shah and J. Pomerantz, *Evaluating and predicting answer quality in community QA*, in *Proc. of SIGIR*, 2010.

[195] A. Shtok, G. Dror, Y. Maarek, and I. Szpektor, *Learning from the past: answering new questions with past answers*, in *Proc. of WWW*, 2012.

[196] M. B. Sinai, N. Partush, S. Yadid, and E. Yahav, *Exploiting social navigation*, *Black Hat Asia* **CoRR:abs/1410.0151** (2015).

[197] N. Snavely, S. M. Seitz, and R. Szeliski, *Photo tourism: Exploring photo collections in 3d*, *ACM ToG* **25** (2006), no. 3.

[198] R. Sommer and V. Paxson, *Outside the closed world: On using machine learning for network intrusion detection*, in *Proc. of IEEE S&P*, 2010.

[199] D. Sounthiraraj, J. Sahs, G. Greenwood, Z. Lin, and L. Khan, *Smv-hunter: Large scale, automated detection of ssl/tls man-in-the-middle vulnerabilities in android apps*, in *Proc. of NDSS*, 2014.

[200] J. Srivastava, R. Cooley, M. Deshpande, and P. N. Tan, *Web usage mining: discovery and applications of usage patterns from Web data*, *SIGKDD Explor. Newsl.* **1** (2000), no. 2 12–23.

[201] N. Srndic and P. Laskov, *Practical evasion of a learning-based classifier: A case study*, in *Proc. of IEEE S&P*, 2014.

[202] J. Stasko and E. Zhang, *Focus+ context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations*, in *Proc. of InfoVis*, 2000.

[203] N. Stefanovitch, A. Alshamsi, M. Cebrian, and I. Rahwan, *Error and attack tolerance of collective problem solving: The darpa shredder challenge*, *EPJ Data Science* **3** (2014), no. 1 1–27.

[204] B. Stone-Gross, R. Stevens, A. Zarras, R. Kemmerer, C. Kruegel, and G. Vigna, *Understanding fraudulent activities in online ad exchanges*, in *Proc. of IMC*, 2011.

[205] C. Strapparava and A. Valitutti, *Wordnet affect: an affective extension of wordnet.*, in *Proc. of LREC*, 2004.

[206] G. Stringhini, C. Kruegel, and G. Vigna, *Detecting spammers on social networks*, in *Proc. of ACSAC*, 2010.

[207] G. Stringhini, G. Wang, M. Egele, C. Kruegel, G. Vigna, H. Zheng, and B. Y. Zhao, *Follow the green: growth and dynamics in twitter follower markets*, in *Proc. of IMC*, 2013.

[208] F. Stutzman, R. Gross, and A. Acquisti, *Silent listeners: The evolution of privacy and disclosure on facebook*, *Journal of Privacy and Confidentiality* **4** (2013), no. 2.

[209] Q. Su and L. Chen, *A method for discovering clusters of e-commerce interest patterns using click-stream data*, *ECRA* **14** (2015), no. 1 1–13.

[210] J. Suler and W. L. Phillips, *The bad boys of cyberspace: Deviant behavior in a multimedia chat community.*, *Cyberpsy., Behavior, and Soc. Networking* **1** (1998), no. 3 275–294.

[211] M. Talasila, R. Curtmola, and C. Borcea, *LINK: location verification through immediate neighbors knowledge*, in *Proc. of MobiQuitous*, 2010.

[212] Y. R. Tausczik and J. W. Pennebaker, *Predicting the perceived quality of online mathematics contributions from users' reputations*, in *Proc. of CHI*, 2011.

[213] K. Thomas, C. Grier, D. Song, and V. Paxson, *Suspended accounts in retrospect: An analysis of twitter spam*, in *Proc. of IMC*, 2011.

[214] K. Thomas, D. Iatskiv, E. Bursztein, T. Pietraszek, C. Grier, and D. McCoy, *Dialing back abuse on phone verified accounts*, in *Proc. of CCS*, 2014.

[215] K. Thomas, D. McCoy, C. Grier, A. Kolcz, and V. Paxson, *Trafficking fraudulent accounts: The role of the underground market in twitter spam and abuse*, in *Proc. of USENIX Security*, 2013.

[216] I.-H. Ting, C. Kimble, and D. Kudenko, *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.

[217] N. Tran, B. Min, J. Li, and L. Subramanian, *Sybil-resilient online content voting*, in *Proc. of NSDI*, 2009.

[218] J. Ugander, B. Karrer, L. Backstrom, and C. Marlow, *The anatomy of the facebook social graph*, *CoRR* **abs/1111.4503** (2011).

[219] C. Vega, "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`.

[220] S. Venkataraman, A. Blum, and D. Song, *Limits of learning-based signature generation with adversaries*, in *Proc. of NDSS*, 2008.

[221] B. Viswanath, A. Post, K. P. Gummadi, and A. Mislove, *An analysis of social network-based sybil defenses*, in *Proc. of SIGCOMM*, 2010.

[222] A. H. Wang, *Don't follow me: Spam detection on twitter*, in *Proc. of SECRYPT*, 2010.

[223] G. Wang, K. Gill, M. Mohanlal, H. Zheng, and B. Y. Zhao, *Wisdom in the social crowd: An analysis of quora*, in *Proc. of WWW*, 2013.

[224] G. Wang, T. Konolige, C. Wilson, X. Wang, H. Zheng, and B. Y. Zhao, *You are how you click: Clickstream analysis for sybil detection*, in *Proc. of USENIX Security*, 2013.

[225] G. Wang, M. Mohanlal, C. Wilson, X. Wang, M. Metzger, H. Zheng, and B. Y. Zhao, *Social turing tests: Crowdsourcing sybil detection*, in *Proc. of NDSS*, 2013.

[226] G. Wang, J. Stokes, C. Herley, and D. Felstead, *Detecting malicious landing pages in malware distribution networks*, in *Proc. of DSN*, 2013.

[227] G. Wang, B. Wang, T. Wang, A. Nika, H. Zheng, and B. Y. Zhao, *Whispers in the dark: Analysis of an anonymous social network*, in *Proc. of IMC*, 2014.

[228] G. Wang, B. Wang, T. Wang, A. Nika, H. Zheng, and B. Y. Zhao, *Defending against sybil devices in crowdsourced mapping services*, in *Proc. of MobiSys*, 2016.

[229] G. Wang, T. Wang, H. Zheng, and B. Y. Zhao, *Man vs. machine: Practical adversarial detection of malicious crowdsourcing workers*, in *Proc. of USENIX Security*, 2014.

[230] G. Wang, C. Wilson, X. Zhao, Y. Zhu, M. Mohanlal, H. Zheng, and B. Y. Zhao, *Serf and turf: crowdturfing for fun and profit*, in *Proc. of WWW*, 2012.

[231] G. Wang, X. Zhang, S. Tang, H. Zheng, and B. Y. Zhao, *Unsupervised clickstream clustering for user behavior analysis*, in *Proc. of CHI*, 2016.

[232] W. Wang, H. Wang, G. Dai, and H. Wang, *Visualization of large hierarchical data by circle packing*, in *Proc. of CHI*, 2006.

[233] X. Wang, J. Zhu, A. Pande, A. Raghuramu, P. Mohapatra, T. Abdelzaher, and R. Ganti, *STAMP: Ad hoc spatial-temporal provenance assurance for mobile users*, in *Proc. of ICNP*, 2013.

[234] D. J. Watts and S. Strogatz, *Collective dynamics of 'small-world' networks*, *Nature* (1998), no. 393 440–442.

[235] J. Wei, Z. Shen, N. Sundaresan, and K.-L. Ma, *Visual cluster exploration of web clickstream data*, in *Proc. of VAST*, 2012.

[236] C. Wilson, B. Boe, A. Sala, K. Puttaswamy, and B. Zhao, *User Interactions in Social Networks and Their Implications*, in *Proc. of EuroSys*, 2009.

[237] J. Wortham, "New social app has juicy posts, all anonymous." NY Times, March, 2014. `http://www.nytimes.com/2014/03/19/technology/new-social-app-has-juicy-posts-but-no-names.html`.

[238] J. Wortham, "Whatsapp deal bets on a few fewer friends." NY Times, February, 2014. `http://www.nytimes.com/2014/02/22/technology/whatsapp-deal-bets-on-a-few-fewer-friends.html`.

[239] T. Xu, Y. Chen, L. Jiao, B. Y. Zhao, P. Hui, and X. Fu, *Scaling microblogging services with divergent traffic demands*, in *Proc. of Middleware*, 2011.

[240] W. Xu, Y. Qi, and D. Evans, *Automatically evading classifiers: A case study on pdf malware classifiers*, in *Proc. of IEEE S&P*, 2016.

[241] C. Yang, R. C. Harkreader, and G. Gu, *Die free or live hard? empirical evaluation and new design for fighting evolving twitter spammers*, in *Proc. of RAID*, 2011.

[242] Y. Yang and J. O. Pedersen, *A comparative study on feature selection in text categorization*, in *Proc. of ICML*, 1997.

[243] Z. Yang, S. Cai, Z. Zhou, and N. Zhou, *Development and validation of an instrument to measure user perceived service quality of information presenting web portals*, *Information & Management* **42** (2005), no. 4 575 – 589.

[244] Z. Yang, C. Wilson, X. Wang, T. Gao, B. Y. Zhao, and Y. Dai, *Uncovering social network sybils in the wild*, in *Proc. of IMC*, 2011.

[245] S. Yardi, D. Romero, G. Schoenebeck, and D. Boyd, *Detecting spam in a twitter network*, *First Monday* **15** (2010), no. 1.

[246] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao, *Sybillimit: A near-optimal social network defense against sybil attacks*, in *Proc. of IEEE S&P*, 2008.

[247] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman, *Sybilguard: defending against sybil attacks via social networks*, in *Proc. of SIGCOMM*, 2006.

[248] J. Zhang, M. S. Ackerman, and L. Adamic, *Expertise networks in online communities: structure and algorithms*, in *Proc. of WWW*, 2007.

[249] L. Zhang, J. Yang, and B. Tseng, *Online modeling of proactive moderation system for auction fraud detection*, in *Proc. of WWW*, 2012.

[250] Y. Zhang, Z. Ge, A. Greenberg, and M. Roughan, *Network anomography*, in *Proc. of IMC*, 2005.

[251] Z. Zhang, L. Zhou, X. Zhao, G. Wang, Y. Su, M. Metzger, H. Zheng, and B. Y. Zhao, *On the validity of geosocial mobility traces*, in *Proc. of HotNets*, 2013.

[252] J. Zhao, Z. Liu, M. Dontcheva, A. Hertzmann, and A. Wilson, *Matrixwave: Visual comparison of event sequence data*, in *Proc. of CHI*, 2015.

[253] E. Zheleva and L. Getoor, *To join or not to join: the illusion of privacy in social networks with mixed public and private user profiles*, in *Proc. of WWW*, 2009.

[254] Z. Zhu and G. Cao, *Toward privacy preserving and collusion resistance in a location proof updating system*, *IEEE Transactions on Mobile Computing* **12** (2013), no. 1 51–64.