Secure and Privacy-Preserving Microblogging Services:

Attacks and Defenses

by

Jinxue Zhang

A Dissertation Presented in Partial Fulfillment
of the Requirement for the Degree
Doctor of Philosophy

Approved May 2016 by the
Graduate Supervisory Committee:

Yanchao Zhang, Chair
Junshan Zhang
Lei Ying
Gail-Joon Ahn

ARIZONA STATE UNIVERSITY

August 2016

ABSTRACT

Microblogging services such as Twitter, Sina Weibo, and Tumblr have been emerging and deeply embedded into people's daily lives. Used by hundreds of millions of users to connect the people worldwide and share and access information in real-time, the microblogging service has also became the target of malicious attackers due to its massive user engagement and structural openness. Although existed, little is still known in the community about new types of vulnerabilities in current microblogging services which could be leveraged by the intelligence-evolving attackers, and more importantly, the corresponding defenses that could prevent both the users and the microblogging service providers from being attacked. This dissertation aims to uncover a number of challenging security and privacy issues in microblogging services and also propose corresponding defenses.

This dissertation makes fivefold contributions. The first part presents the *social botnet*, a group of collaborative social bots under the control of a single botmaster, demonstrate the effectiveness and advantages of exploiting a social botnet for spam distribution and digital-influence manipulation, and propose the corresponding countermeasures and evaluate their effectiveness. Inspired by Pagerank, the second part describes *TrueTop*, the first sybil-resilient system to find the top-K influential users in microblogging services with very accurate results and strong resilience to sybil attacks. TrueTop has been implemented to handle millions of nodes and 100 times more edges on commodity computers. The third and fourth part demonstrate that microblogging systems' structural openness and users' carelessness could disclose the later's sensitive information such as home city and age. *LocInfer*, a novel and lightweight system, is presented to uncover the majority of the users in any metropolitan area; the dissertation also proposes *MAIF*, a novel machine learning framework that leverages public content and interaction information in microblogging services to infer users' hidden

ages. Finally, the dissertation proposes the first privacy-preserving social media publishing framework to let the microblogging service providers publish their data to any third-party without disclosing users' privacy and meanwhile meeting the data's commercial utilities. This dissertation sheds the light on the state-of-the-art security and privacy issues in the microblogging services.

To My Family.

ACKNOWLEDGMENTS

During past five years, it is a tremendous blessing to encounter so many great people who have directly or indirectly make this whole thing happened.

First of all, I would like to thank my advisor Dr. Yanchao Zhang and his family. He is always a bright role model for me as a researcher and advisor. All these work could not make without his endless passion for seeking and solving the challenging and meaningful research problems, his rigorousness to pursue the details, his openness to encourage us to explore new and unknown domain, his braveness to face and solve the difficult problems, his great patience for discussing with all our group members, his huge tolerance for our failures and attempts, and his continuous support from every aspect.

I also give my sincere appreciations to my committee members, Dr. Junshan Zhang, Dr. Lei Ying, and Dr. Gail-Joon Ahn. They sacrifices their precious time and effort to give me many constructive suggestions starting from the first year of my graduate period.

Not only the professors from Arizona State University, this dissertation also benefits from many other external scholars. Dr. Guanhua Yan from Binghamton University provided many insightful discussions for the Chapter 2. Dr. Xia Ben Hu from Texas A&M University contributed the technical idea of Chapter 5 and helped to initialize the problem formation of Chapter 6. Without them, this dissertation will be much less interesting and comprehensive.

We act as a team–I thank for many members from ASU Cyber & Network Security Group (CNSG). Dr. Rui Zhang helped me to settle down in ASU back in 2011 summer and experienced with great patience my initial struggle in research. During past five years, he has been always reachable for any discussion, and in-hand coaching in writing and research methodologies. Over half of this dissertation has been polished

by him. I also thank Jingchao Sun, Xiaocong Jin, Yimin Chen, Tao Li, and Xin Yao for their useful discussion in each group meeting, the earnest support and help in many emergent situations, and most importantly, the mental encouragement in every single day in the lab.

I also thank many friends in GPCCC and ASU campus from China, U.S., Korea, and India. Through prays, conversations, team work and play, they always recharged me and gave me peace, support, and patience for every challenge in both work and life. It is a great treasure to know them and befriend with them. Here I give the most special thanks to my girlfriend Joyce Hu for the continuous and unselfish comfort and support during the formation of this dissertation. Her coming to my life is the most significant gift from God.

Finally, I gave my inexpressible gratefulness to my family in China. They always trust me, support me, and comfort me, and have made huge sacrifice during past five to ten years. This dissertation is made to honor them.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Chapter 1

INTRODUCTION

Microblogging services such as Twitter, Sina Weibo, and Tumblr have been emerging and deeply embedded into people's daily lives. As of September 2015, Twitter—the most popular microblogging system in the world—has 320 million monthly active users. People have been using microblogging systems in social networking, massive information campaigns, public relationships, political campaigns, pandemic and crisis situations, business marketing, crowdsourcing, and many other public/private contexts.

Used by hundreds of millions of users to connect the people worldwide and share and access information in real-time, the microblogging service has also became the target of malicious attackers due to its massive user engagement and structural openness. Although existed, little is still known in the community about new types of vulnerabilities in current microblogging services which could be leveraged by the intelligence-evolving attackers, and more importantly, the corresponding defenses that could prevent both the users and the microblogging service providers from being attacked.

In this dissertation, we aim to uncover a number of challenging security and privacy issues in microblogging services and also propose corresponding defenses. The rest of this report is structured as follows.

Chapter 2 studies the consequences of and corresponding defenses against social botnets in the context of microblogging services. Specifically, Online social networks (OSNs) are increasingly threatened by *social bot*s which are software-controlled OSN accounts that mimic human users with malicious intentions. A *social botnet* refers to a

group of social bots under the control of a single *botmaster*, which collaborate to conduct malicious behavior while mimicking the interactions among normal OSN users to reduce their individual risk of being detected. We demonstrate the effectiveness and advantages of exploiting a social botnet for spam distribution and digital-influence manipulation through real experiments on Twitter and also trace-driven simulations. We also propose the corresponding countermeasures and evaluate their effectiveness. Our results can help help OSNs improve their bot(net) detection systems.

Chapter 3 further presents a novel defend scheme against the social botnets (or sybil users) in the application of finding top-K influential users in the microblogging services. To start with, influential users have great potential for accelerating information dissemination and acquisition on Twitter. How to measure the influence of Twitter users has attracted significant academic and industrial attention. Existing influence measurement techniques are vulnerable to sybil users that are thriving on Twitter. Although sybil defenses for online social networks have been extensively investigated, they commonly assume unique mappings from human-established trust relationships to online social associations and thus do not apply to Twitter where users can freely follow each other. This chapter presents TrueTop, the first sybil-resilient system to measure the influence of Twitter users. TrueTop is rooted in two observations from real Twitter datasets. First, although non-sybil users may incautiously follow strangers, they tend to be more careful and selective in retweeting, replying to, and mentioning other users. Second, influential users usually get much more retweets, replies, and mentions than non-influential users. Detailed theoretical studies and synthetic simulations show that TrueTop can generate very accurate influence measurement results with strong resilience to sybil attacks.

Chapter 4 investigates one privacy threat on microblogging services, namely finding the majority users in any metropolitan area. Specifically, most Twitter users do

not disclose their locations due to privacy concerns. Although inferring the location of an individual Twitter user has been extensively studied, it is still missing to effectively find the majority of the users in a specific geographical area without scanning the whole Twittersphere, and obtaining these users will result in both positive and negative significance. In this chapter, we propose LocInfer, a novel and lightweight system to tackle this problem. LocInfer explores the fact that user communications in Twitter exhibit strong geographic locality, which we validate through large-scale datasets. Based on the experiments from four representative metropolitan areas in U.S., LocInfer can discover on average 86.6% of the users with 73.2% accuracy in each area by only checking a small set of candidate users. We also present a countermeasure to the users highly sensitive to location privacy and show its efficacy by simulations.

Chapter 5 discloses another privacy issue on Twitter, namely inferring users' ages. The age information of microbloggers can be very useful for many applications such as viral marketing and social studies/surveys. Current microblogging systems, however, have very sparse age information. In this chapter, we present MAIF, a novel framework that explores public content and interaction information in microblogging systems to explore the hidden ages of microbloggers. We thoroughly evaluate the accuracy of MAIF with a real-world dataset with 54,879 Twitter users. Our results show that MAIF can achieve up to 81.38% inference accuracy and outperforms the state of the art by 9.15%. We also discuss some countermeasures to alleviate the possible privacy concerns caused by MAIF.

Chapter 6 considers how to prevent the microbloggers' privacy disclosure mentioned above. We first identify a text-based user-linkage attack on current social media data publishing practices, in which the real users of anonymous IDs in a published dataset can be pinpointed based on the users' unprotected text data. Then we

propose a framework for differentially privacy-preserving social media data publishing for the first time in literature. Within our framework, social media data service providers can publish perturbed datasets to provide differential privacy to social media users while offering high data utility to social media data consumers. Our differential privacy mechanism is based on a novel notion of ϵ-text indistinguishability, which we propose to thwart the text-based user-linkage attack. Extensive experiments on real-world and simulated datasets confirm that our framework can enable high-level differential privacy protection and also high data utility at the same time.

We summarize our work and present several future work in Chapter 7.

Chapter 2

THE RISE OF SOCIAL BOTNETS: ATTACKS AND COUNTERMEASURES

2.1 Introduction

Online social networks (OSNs) are increasingly threatened by *social bot*s (52) which are software-controlled OSN accounts that mimic human users with malicious intentions. For example, according to a May 2012 article in Bloomberg Businessweek, [1] as many as 40% of the accounts on Facebook, Twitter, and other popular OSNs are spammer accounts (or social bots), and about 8% of the messages sent via social networks are spams, approximately twice the volume of six months ago. There have been reports on various attacks, abuses, and manipulations based on social bots (51), such as infiltrating Facebook (26) or Twitter (23; 53), launching spam campaign (57; 62; 132), and conducting political astroturf (119; 120).

A *social botnet* refers to a group of social bots under the control of a single *botmaster*, which collaborate to conduct malicious behavior while mimicking the interactions among normal OSN users to reduce their individual risk of being detected. For example, social bots on Twitter can follow others and retweet/answer others' tweets. Since a skewed following/followers (FF) ratio is a typical feature for social bots on Twitter (141), maintaining a balanced FF ratio in the social botnet makes it much easier for individual bots to escape detection. Creating a social botnet is also fairly easy due to the open APIs published by OSN providers. For example, we successfully created a network of 1,000 accounts on Twitter with $57 to purchase 1,000 accounts instead of manually creating them.

[1] http://www.businessweek.com/articles/2012-05-24/likejacking-spammers-hit-social-media

Despite various studies (58; 151; 59) confirming the existence of social botnets, neither have the greater danger from social botnets been unveiled nor have the countermeasures targeted on social botnets been proposed. In this chapter, we first report two new social botnet attacks on Twitter, one of the most popular OSNs with over 302M monthly active users as of June 2015 and over 500M new tweets daily. Then we propose two defenses on the reported attacks, respectively. Our results help understand the potentially detrimental effects of social botnets and shed the light for Twitter and other OSNs to improve their bot(net) detection systems. More specifically, this chapter makes the following contributions.

Firstly, we demonstrate the effectiveness and advantages of exploiting a social botnet for *spam distribution* on Twitter. This attack is motivated by that Twitter currently only suspends the accounts that originate spam tweets without punishing those retweeting spam tweets (11). If the social botnet is organized as a retweet tree in which only the root originates spam tweets and all the others merely retweet spams, all the social bots except the root bot can escape suspension. Given a set of social bots, we formulate the formation of the retweeting tree as a multi-objective optimization problem to minimize the time taken for a spam tweet to reach a maximum number of victim Twitter users at the lowest cost of the botmaster. Since the optimization is NP-hard, we give a heuristic solution and confirm its efficacy with real experiments on Twitter and trace-driven simulations.

Secondly, we show that a social botnet can easily manipulate the *digital influence* (30; 143) of Twitter users, which has been increasingly used in ad targeting (144; 6), customer-service improvement (75), recruitment (66), and many other applications. This attack stems from the fact that almost all existing digital-influence tools such as Klout, Kred, and Retweet Rank, measure a user's digital influence exclusively based

on his [2] interactions with others users on Twitter. If social bots collaborate to manipulate the interactions of target Twitter users, they could effectively manipulate the victims' digital influence. The efficacy of this attack is confirmed by real experiments on Twitter.

Finally, we propose two countermeasures to defend against the two reported attacks, respectively. To defend against the botnet-based spam distribution, we maintain a spam score for each user and update the score whenever the corresponding user retweets a spam. The user is suspended if his spam score exceeds a predefined threshold. To defense against the botnet-based influence manipulation attacks, we propose to find sufficient credible users and only use the interactions originated from these credible users for digital-influence measurement. Moreover, based on the measurement in Section §2.4, we design a new model to compute the influence score which is resilient to the manipulation from a single credible social bot. We confirm the effectiveness of both defenses via detailed simulation studies driven by real-world datasets.

The rest of the chapter is organized as follows. §2.2 introduces the construction of a social botnet on Twitter. §2.3 and §2.4 show the efficacy and merits of using the social botnet for spam distribution and digital-influence manipulation, respectively. §2.5 details and evaluates two countermeasures. §2.6 discusses the related work. §3.7 concludes this chapter.

2.2 Building a Social Botnet on Twitter

In this chapter, we focus on networked social bots in Twitter, so we first outline the Twitter basics to help illustrate our work. The readers familiar with Twitter can safely skip this paragraph without any loss of continuity. Unlike Facebook, the

[2] No gender implication.

social relationships on Twitter are unidirectional by users *following* others. If user A follows user B, A is B's *follower*, and B is A's *friend*. In most cases, a user does not need prior consent from another user whom he wants to follow. Twitter also allows users to control who can follow them, but this feature is rarely used. In addition, users can choose to *unfollow* others and *block* their selected followers. A Twitter user can send text-based posts of up to 140 characters, known as *tweet*s, which can be read by all its followers. Tweets can be public (the default setting) and are visible to anyone with or without a Twitter account, and they can also be protected and are only visible to previously approved Twitter followers. A *retweet* is a re-posting of someone else's tweet. A user can retweet the tweets of anyone he follows or does not follow, and his retweets can be seen by all his followers. Moreover, a user can *reply* to a tweet and ensure that specific users can see his posts by *mentioning* them via inserting "@username" for every specific user into his posts. Finally, each user has a *timeline* which shows all the latest tweets, retweets, and replies of his followers.

We construct a social botnet on Twitter consisting of a botmaster and a number of social bots which are legitimate Twitter accounts. Twitter accounts can be manually created or purchased at affordable prices. For example, we bought 1,000 Twitter accounts with $57 from some Internet sellers for experimental purposes only. The botmaster is in the form of a Java application, which we developed from scratch based on the OAuth protocol (7) and open Twitter APIs. It could perform all the Twitter operations on behalf of all social bots to make the bots look like legitimate users.

2.3 Social Botnet for Spam Distribution

2.3.1 Why the Social Botnet for Spam Distribution?

As the popularity of Twitter rapidly grows, spammers have started to distribute spam tweets which can be broadly defined as unwanted tweets that contains malicious URLs in most cases or occasionally malicious texts (62; 132; 35). According to a study in 2010 (62), roughly 8% of the URLs in tweets are malicious ones that direct users to scams/malware/phishing sites, and about 0.13% of the spam URLs will be clicked. Given the massive scale of Twitter, understanding how spam tweets are distributed is important for designing effective spam defenses as we will demonstrate in § 2.5.

The simplest method for spam distribution is to let social bots distribute spam tweets independently from each other, which we refer to as the *independent method*. In particular, the botmaster can instruct every bot to directly post spam tweets which can be seen by all its followers. According to the Twitter rules, [3] the accounts considered as spam originators will be permanently suspended. Since there are sophisticated techniques such as (131; 95) detecting malicious URLs, this independent approach may subject almost all social bots to permanent suspension in a short time window.

A more advanced method, which we propose and refer to as the *botnet method*, is to exploit the fact that Twitter currently only suspends the originators of spam tweets without punishing their retweeters. In the simplest case, the botmaster forms a single *retweeting tree*, where every bot is associated with a unique vertex and is followed by its children bots. Then only the root bot originates spam tweets, and all the others simply retweet the spam tweets from their respective parent. Given the same set of social bots, both methods can distribute spam tweets to the same set

[3] `http://support.twitter.com/articles/18311\#`

of non-bot Twitter users, but only the root bot will be suspended under the botnet method. Obviously, the botnet method is economically beneficial for the botmaster because it involves non-trivial human effort or money to create a large social botnet.

We use an experiment on Twitter to validate our conjecture for the independent method. Our experiment uses three different social botnets with each containing 100 bots. The experiment proceeds in hours. At the beginning of every hour, every bot in the same botnet almost simultaneously posts a spam tweet comprising two parts. The first part is different from every bot and randomly selected from the list of tweets returned after querying "music," while the second part is an identical malicious URL randomly selected from the Shalla's blacklists (`http://www.shallalist.de/`) and shortened using the bitly service (`http://bitly.com`) for use on Twitter. We find that all the bots in the three botnets are suspected in two, five, and six hours. Based on this experiment, we can safely conjecture that the independent method will cause most bots in a larger botnet to be suspended in a short period, thus putting the botmaster at serious economic disadvantage.

We use a separate set of experiments to shed light on the advantage of the botnet method. In this experiment, we first use 111 bots to build a full 10-ary tree of depth two, i.e., each node except the leaves has exactly 10 children. The experiment proceeds in hourly rounds repeatedly on these 111 bots. At the beginning of every hour of the first round, the root bot posts a spam tweet, while all its descendants merely retweet the spam tweet after a small random delay. Then we replace the suspended bot by a random bot alive from the same network, re-organize the bot tree, and start the next round. We totally run the experiments for five rounds, in each of which only the root bot is suspended after six hours on average, and all other bots who just retweet the spams (with five times) remain alive. To check whether the bots will be suspended by retweeting more spams, we reduce the spamming frequency from one per hour to

one per day, and repeat the experiment for ten more rounds, and all the retweeting bots were still alive at the end of the experiment. In addition, we use the similar methodology to test three other different botnets of 2, 40, and 100 bots, respectively, and obtain the similar results.

It has been very challenging in the research community to conduct experimental studies about the attacks on online social networks and also the corresponding countermeasures. In the experiments above, we have to control the social bots to post malicious URLs to evaluate Twitter's suspension policy, which may harm benign users. To minimize the negative impact on the legitimate users, we adopted a methodology similar to (26; 132; 101; 152). Specifically, none of the purchased accounts followed any legitimate user and thus were very unlikely to be followed by legitimate users, which greatly reduced the possibility of the posted spams being viewed by legitimate users. In addition, we deleted every spam tweet immediately after the experiment to further avoid it being clicked by legitimate users. Our experiments clearly show that Twitter has a much more strict policy against posting original spam tweets than retweeting spam tweets.

2.3.2 Optimal Social Botnet for Spam Distribution

§2.3.1 motivates the benefits of using the social botnet for spam distribution on Twitter. Given a set of social bots, what is the optimal way for spam distribution? We give an affirmative answer to this important question in this section.

Problem Setting and Performance Metrics

We consider a botnet $\mathcal{V}$ of n bots, where each bot $i \in [1, n]$ can be followed by other bots and also other Twitter users outside the botnet (called non-bot followers hereafter). Let $\mathcal{F}_i$ denote the non-bot followers of bot i. Note that $\mathcal{F}_i \cap \mathcal{F}_j$ may

be non-empty ($\forall i \neq j$), meaning that any two bots may have overlapping non-bot followers. We further let $\mathcal{F} = \bigcup_{i=1}^{n} \mathcal{F}_i$. How to attract non-bot followers for the bots is related to social engineering (37) and orthogonal to the focus of this chapter. Note that it is very easy in practice for a bot to attract many non-bot followers, as shown in (23; 26; 59; 151).

The botmaster distributes spam tweets along one or multiple retweeting trees, and the vertices of every retweeting tree corresponds to a disjoint subset of the n bots. In addition, every bot in a retweeting tree is followed by its children. As discussed, the root of every retweeting tree will originate spam tweets, which will appear in the Twitter timeline of its children bots and then be retweeted. The distribution of a particular spam tweet finishes until all the bots on all the retweeting trees either tweet or retweet it once and only once.

Given a set $\mathcal{V}$ with n bots and $\mathcal{F}$, we propose three metrics to evaluate the efficacy of botnet-based spam distribution.

- *Coverage*: Let $\mathcal{C}$ denote the non-bot receivers of a given spam tweet and be called the *coverage set*. The coverage of spam distribution is then defined as $\frac{|\mathcal{C}|}{|\mathcal{F}|} \in [0, 1]$.

- *Delay*: We define the delay of spam distribution, denoted by τ, as the average time for each user in $\mathcal{C}$ to see a given spam tweet since it is generated by the root bot. A user may follow multiple bots and thus see the same spam tweet multiple times, in which case only the first time is counted.

- *Cost*: We use $|\mathcal{S}|$ and $|\tilde{\mathcal{S}}|$ to define the cost of spam distribution, where $\mathcal{S}$ denotes the indices of suspended bots after distributing a given spam, and $\tilde{\mathcal{S}}$ denotes the set of non-bot followers will be lost due to the suspension of $\mathcal{S}$, i.e., $\tilde{\mathcal{S}} = \mathcal{C} \setminus (\bigcup_{i \in \mathcal{V} \setminus \mathcal{S}} \mathcal{F}_i)$.

The above metrics motivates three design objectives. First, we obviously want to maximize the coverage to be one, which happens when all the n bots participate in spam distribution by belonging to one retweeting tree. Second, many malicious URLs in spam tweets are hosted on compromised servers and will be invalidated once detected, and Twitter will remove spam tweets as soon as they are identified. It is thus also important to minimize the delay. Finally, since it incurs non-trivial human effort or money to create bots and attract followers for them, it is critical to minimize the cost as well.

Design Constraints

A major design challenge is how to circumvent Twitter's suspension rules [4] that are evolving in accordance with changing user (mis)behavior. We classify the suspension rules into *strict* and *loose* ones. Violators of strict rules will be immediately suspended. The strict rule most relevant to our work is that the users originate spam tweets containing malicious URLs will be suspended. In contrast, a violator of loose rules will initially become suspicious and later be suspended if his violations of related loose rules exceed some unknown threshold Twitter defines and uses internally. Examples of loose rules include repeatedly posting others' tweets as your own or the same tweet, massively following/unfollowing people in a short time period, etc. In addition, the research community have discovered many useful loose rules for spam-tweet detection such as those in (78; 82; 125; 123; 131; 150; 157) which are likely to be or have been adopted by Twitter into their evolving suspension-rule list. As discussed, we use the botnet method for spam distribution in order to largely circumvent this strict rule. In the following, we introduce five design constraints related to some loose rules we consider most relevant. By following these constraints, the social

[4] `http://support.twitter.com/articles/18311#`

bots can cause much less suspicion to Twitter and thus are much less likely to be suspended.

1. The maximum height of a retweeting tree is $K = 10$ according to (78). Hence we claim that any spam tweet will not be retweeted more than 10 times.

2. A bot only retweets the spam tweets posted by its parent bot on the retweeting tree it follows, as retweeting the tweets from non-followed users is known to be effective in detecting spam tweets (123).

3. Any spam tweet from an arbitrary bot will be retweeted by at most $100r$ percent of its followers. As r approaches one, the bot will become increasingly suspicious according to community-based spam detection algorithms (56; 35). Recall that the followers of any bot $i \in [1, n]$ comprise other bots and also non-bot users (i.e., $\mathcal{F}_i$). Note that non-bot followers rarely retweet spam tweets in practice, but we require all bot followers to retweet spam tweets. Then bot i can have no more than $\lceil \frac{r|\mathcal{F}_i|}{1-r} \rceil$ bot followers.

4. The retweeting lag at any hop $j \in [1, K]$ is a random variable t_i which follows a hop-wise statistical distribution according to (78), as it is quite abnormal for a user to immediately retweet a post once seeing it. Here the retweeting lag is defined as the time elapsed when a bot sees a spam tweet until it retweets it.

5. The social bots within the first M hops will be suspended once Twitter finds that they are involved in (re)tweeting a spam tweet. This constraint is motivated by recent findings (151) that spammer accounts on Twitter tend to be connected and clustered by mutual followings. It is thus reasonable to assume that Twitter either have been utilized or will soon utilize these research findings to suspend the accounts involved in distributing a spam tweet within the first $M > 0$ hops. After introducing this constraint, we relax the third one by allowing arbitrary topology in the first M hops because all of its bots will be suspended.

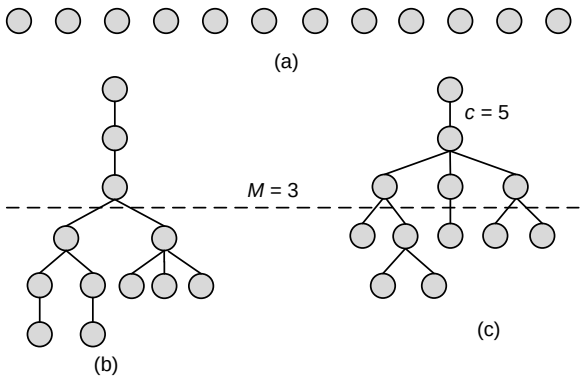


Figure 2.1: Exemplary Retweeting Trees With 12 Bots, Where $M = 3$ And the Botmaster's Suspension Budget is $c = 5$.

Problem Formulation

Give the above design objectives and constraints, we now attempt to formulate botnet-based spam distribution as an optimization problem. The major challenge lies in the infeasibility of simultaneously achieving the maximum coverage, the minimum delay, and the minimum cost. Fig. 2.1 shows an example with 12 bots and $M = 3$, and we assume that every bot has the same number of non-bot followers. In one extreme shown in Fig. 2.1(a), we can minimize the delay τ by letting every bot be a root bot, but the cost is obviously the maximum possible because all the bots will be suspended. In the other extreme shown in Fig. 2.1(b), we can form a single retweeting tree with exactly three bots within the first three hops, in which case we can achieve the minimum possible cost, but the achievable delay will always be larger than that in the first case no matter how the retweeting tree beyond three hops is formed. In addition, we assume for the second case that the retweeting tree can include all the 12 bots, leading to the same coverage of one as in the first case. If there are too many bots, however, some of them may not be able to be incorporated into the retweeting tree due to the first and third constraints, and the resulting coverage will be smaller than that of the first case.

To deal with the above challenge, assume that the botmaster has a suspension budget $c \in [M, n]$ bots, referring to the maximum number of suspended bots it can tolerate. Note that the more bots in the first M hops, the more non-bot followers in $\mathcal{F}$ closer to the root bot which can receive a given spam tweet in shorter time, and thus the smaller the delay. Under the budget constraint, the minimum delay can hence be achieved only when there are exactly c bots within the first M hops, as shown in Fig. 2.1(c) with $c = 5$.

What is the optimal way to form a retweeting tree as in Fig. 2.1(c) given the cost, coverage, and delay requirements? Recall that the cost is defined by $|\mathcal{S}|$ and $|\tilde{\mathcal{S}}|$. Since $|\mathcal{S}| = c$ under the budget constraint, we just need to minimize $|\tilde{\mathcal{S}}|$. To mathematically express the cost and coverage requirements, we let $\{\mathcal{V}_k\}_{k=1}^{K}$ denote K disjoint subsets of the bot indices $\{1, \ldots, n\}$, where $K = 10$ is the maximum height of the retweeting tree (see Constraint 1), $\mathcal{V}_k$ denote the bot indices at level k of the retweeting tree, and $\bigcup_{k=1}^{K} \mathcal{V}_i \subseteq \{1, \ldots, n\}$. If the optimal retweeting tree eventually found is of depth $K^* < K$, the sets $\{\mathcal{V}_k\}_{k=K^*+1}^{K}$ will all be empty. Recall that $\mathcal{F}_i$ denotes the set of non-bot followers of bot $i \in [1, n]$ and that $\mathcal{F} = \bigcup_{i=1}^{n} \mathcal{F}_i$. Then we have $\tilde{\mathcal{S}} = \mathcal{C} \setminus (\bigcup_{i \in \mathcal{V}_k, k \in [M+1, K^*]} \mathcal{F}_i)$ and the coverage set $\mathcal{C} = \bigcup_{i \in \mathcal{V}_k, k \in [1, K]} \mathcal{F}_i \subseteq \mathcal{F}$ and need to maximize $|\mathcal{C}|$. Since $\tilde{\mathcal{S}} \subseteq \mathcal{C}$, we can combine the cost and coverage requirements into a single metric $\frac{|\tilde{\mathcal{S}}|}{|\mathcal{C}|}$ and then attempt to minimize it.

It is a little more complicated to derive the delay. As discussed, a non-bot user may follow multiple bots at different levels, in which case it is considered a follower on the lowest level among those. Let Φ_k denote the set of non-bot followers at $k \in [1, K]$. It follows that $\Phi_1 = \mathcal{F}_i$ $(i \in \mathcal{V}_1)$ and $\Phi_k = \bigcup_{i \in \mathcal{V}_k} \mathcal{F}_i - \bigcup_{l=1}^{k-1} \Phi_l$ for $k \in [1, K]$. According to Constraint 4, it takes $\sum_{j=1}^{k} t_j$ for a spam tweet to reach level-k non-bot followers, where t_j denotes the retweeting lag of hop $j \in [1, K]$, and $t_1 = 0$. Since there are totally $|\Phi_k|$ non-bot followers at level k and $|\mathcal{C}|$ non-bot followers across all levels, we

can compute the delay as

$$\tau = \frac{1}{|\mathcal{C}|} \sum_{k=1}^{K} \sum_{j=1}^{k} t_j |\Phi_k| \ .$$

Finally, we reduce the three-objective optimization problem to the following single-objective minimization problem.

$$\begin{aligned} \min \quad & f(\{\mathcal{V}_k\}_{k=1}^{K}) = \alpha\beta \frac{|\tilde{\mathcal{S}}|}{|\mathcal{C}|} + (1-\alpha)\tau \\ \text{s.t.} \quad & \bigcup_{k=1}^{K} \mathcal{V}_i \subseteq \{1, \ldots, n\} \\ & \mathcal{V}_i \cap \mathcal{V}_j = \phi, \forall i \neq j \in [1, K] \\ & |\bigcup_{k=1}^{M} \mathcal{V}_i| \leq c \\ & \sum_{i \in \mathcal{V}_k} \lceil \frac{r|\mathcal{F}_i|}{1-r} \rceil \geq |\mathcal{V}_{k+1}|, \forall k \in [M-1, K-1] \end{aligned} \tag{2.1}$$

We have two remarks here. First, $\alpha \in [0, 1]$ is a adjustable weight that reflects the relative importance of coverage and delay, and β is a fixed scaling factor to unify two different objective units. Second, the last constraint is due to the aforementioned third design constraint.

The above optimization problem can be viewed as a variation of classical set partition problem (SPP), which is NP-hard. In what follows, we introduce a heuristic approximation solution by constructing a collection of disjointed subsets $\{\mathcal{V}_k\}_{k=0}^{K}$ from the botnet set.

Heuristic Solution

Our intuition is to use all the budget c and fill the first M hops of the retweeting trees with the bots having the lowest suspension cost in terms of the number of lost non-bot followers. We then recursively place the bots with the highest number of

non-bot followers from level $M+1$ to the last level, in order to reduce the average latency as well as cost.

To begin with, our approximation is built on the solutions to the traditional maximum coverage problem (or MaxCover) and the minimum coverage problem (MinCover), which is to select k sets from a collection of subsets of a ground set so that the their union is maximized (34) or minimized, respectively. MaxCover and MinCover problems are both NP-hard and have greedy approximation algorithms by iteratively selecting the maximum or minimum subset after extracting the selected elements, respectively.

Our solution consists of the following two steps. First, given the budget c for $\mathcal{S}$ and that all the followers in $\tilde{\mathcal{S}}$ will be lost because of the suspension, we minimize the objective $|\tilde{\mathcal{S}}|$ by using MinCover to choose c bots as $\tilde{\mathcal{S}}$ with the minimum total number of non-bot followers. In doing so, we can determine the union of the bot set for the first M level. The bot subset in each level will be determined later. Here we just assume that $\tilde{\mathcal{S}}$ has been divided into the M groups, each corresponding to the bots in one of the first M levels. Second, we construct $\{\mathcal{V}_k\}_{k=M+1}^{K}$ to greedily increase the coverage C and at the same time lower the average delay T. Specifically, assuming that we have known $\mathcal{V}_M$, to determine $\mathcal{V}_{M+1}$, we first set the cardinality of $\mathcal{V}_{M+1}$ be equal to $\sum_{i\in\mathcal{V}_k}\lceil\frac{r|\mathcal{F}_i|}{1-r}\rceil$ according to the last constraint in (2.1) and then use MaxCover to choose a subset of $|\mathcal{V}_{M+1}|$ bots from the remaining bot set with the maximum number of non-bot followers. We repeat this greedy selection for every level $k = M+2, \ldots, K$.

The remaining problem is how to partition $\tilde{\mathcal{S}}$ into M subsets, each corresponding to one of the first M levels. A heuristic observation here is that we need to maximize $|\mathcal{V}_M|$, as the more non-bot followers of the social bots in the Mth level, the more social bots in level $M+1$ and subsequent levels, and also the lower average delay according to

the last constraint in (2.1). Given the budget $|\tilde{\mathcal{S}}| = c$, we obtain $|\mathcal{V}_M|_{max} = c - M + 1$ when the retweeting forest has a single tree whose first $M - 1$ levels form a straight line, as shown in Fig. 2.1(b). The bots in the Mth level is then determined by using MaxCover to choose the $c - M + 1$ bots from $\tilde{\mathcal{S}}$ with the maximum number of non-bot followers.

To determine the level for each of the remaining $M-1$ bots, we sort the remaining $M - 1$ bots in $\tilde{\mathcal{S}}$ in the descending order according to the number of their non-bot followers and assign them to the corresponding level, e.g., the bot with the highest number of non-bot followers will be assigned to the first level. Note that it is possible that after we maximizing the number of bots at the Mth level, the remaining bots are less than the allowed on the Mth level, so the $(M + 1)$-th is not full. To further reduce the average delay in such cases, we move the exceeding bots in the Mth level to the first level.

After determining $\{\mathcal{V}_i\}_{i=1}^{K}$ from the social-bot set $\{1, \ldots, n\}$, we can then build the final retweeting forest (tree). Specifically, the number of retweeting trees is equal to the cardinality of $\mathcal{V}_1$, which is one if the $(M + 1)$-th level is full or greater than one otherwise. We then randomly choose one social bot from $\mathcal{V}_1$ to be the root of the retweeting tree with more than one level, which is followed by the bots from the second to Mth level determined by $\mathcal{V}_2$ to $\mathcal{V}_M$, respectively. Finally, we build the level from $k = M + 1$ to K by selecting certain number of social bots from $\mathcal{V}_k$ according the last constraint in Eq. (2.1).

2.3.3 Trace-driven Evaluation

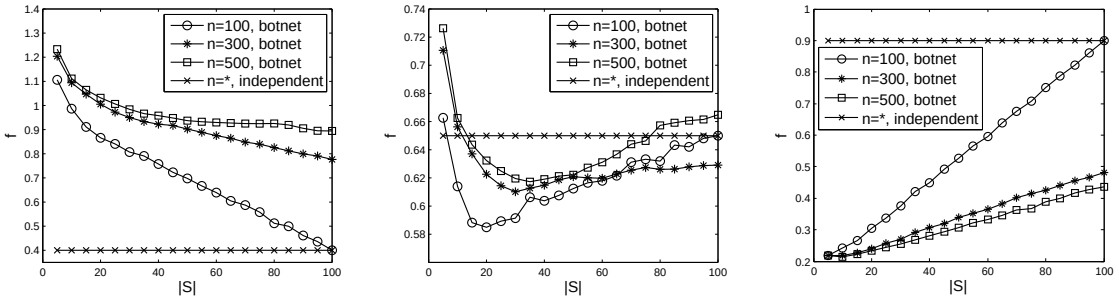
We conduct trace-driven simulations to compare the performance of spam distribution using the independent and botnet methods, as well as evaluating the tradeoffs among the multiple goals in the botnet method.

The evaluation for independent bots is straightforward. In particular, given the bot set $\mathcal{V}$ with $|\mathcal{V}| = n$, we place all the bots in the first level which will be suspended completely. We then have $\mathcal{C} = \tilde{\mathcal{S}} = n$, and $\tau = 0$. The single objective in Problem (2.1) is thus $f = \alpha$.

To evaluate the botnet method, we set up the simulations according to existing measurement data and heuristics. We set $K = 10$, and $t_1 = 0, t_i = 0.5i$ hour for $i = 2, \dots K$ according to (78). To build $\{\mathcal{F}_i\}_{i=1}^n$, we generate $|\mathcal{F}_i|$ according to the Gaussian distribution with $\mu = 32$ as the average number of followers in the dataset of (78). We also set the variance to $\sigma^2 = 5$, generate a legitimate follower set $\mathcal{F}$ with $|\mathcal{F}| = 6000$, [5] and randomly choose $|\mathcal{F}_i|$ followers from the set $\mathcal{F}$ for each bot i. In addition, according to (151), the average path length of the spammer community is 2.60, so we set $M = 3$ to suspend the bots in the first three hops of $\mathcal{F}$. Finally, we set β to one and the retweeting ratio $r = 0.2$. Due to space constraints, we focus on on the impact of α, c, and n and do not report the impact of $|\mathcal{F}|$, σ^2, r, M, or β in this chapter.

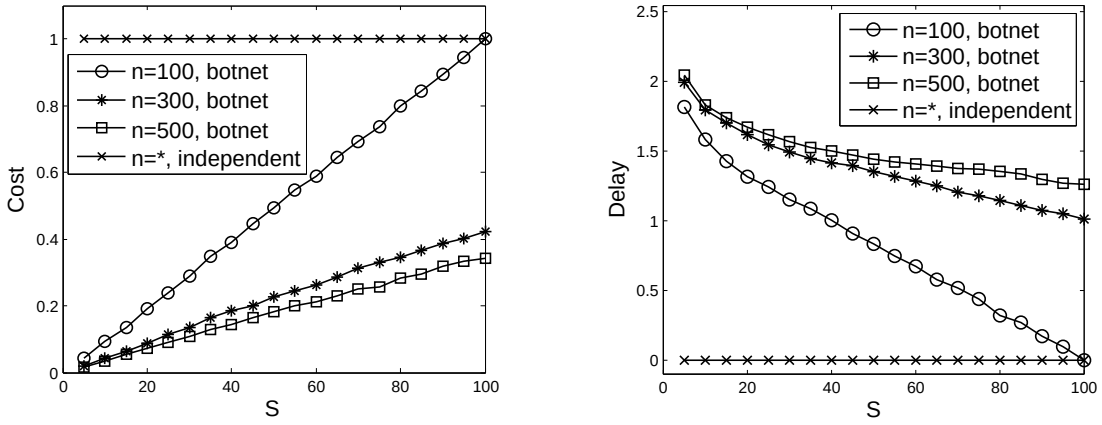
Fig. 2.2 compares the independent and botnet methods using the objective function f with different weights α. As stated before, f is simply equal to α for the independent case because $\tau = 0$ and $\tilde{\mathcal{S}} = \mathcal{C}$. For the botnet method, the objective f is the weighted sum of $\frac{|\tilde{\mathcal{S}}|}{|\mathcal{C}|}$ and the delay τ. When α is small, τ has higher impact on f than $\frac{|\tilde{\mathcal{S}}|}{|\mathcal{C}|}$, while when α is large, $\frac{|\tilde{\mathcal{S}}|}{|\mathcal{C}|}$ will dominate f. Specifically, we can see from Fig. 2.2a that when $\alpha = 0.4$, the independent method outperforms the botnet method with smaller f. However, as shown in Fig. 2.2b, when α increases to 0.65, the botnet method can achieve lower f than the independent method does. This trend is more obvious for the same reason in Fig. 2.2c where $\alpha = 0.9$.

[5] For the Gaussian distribution with $\mu = 32$ and $\sigma^2 = 5$, the probability for generating a negative $|\mathcal{F}_i|$ is negligible.



(a) $\alpha = 0.4$ (b) $\alpha = 0.65$ (c) $\alpha = 0.9$

Figure 2.2: Performance Comparison of Independent and Botnet Methods in Spam Distribution at Different αs in Terms of the Single Objective f.



(a) #lost legitimate followers (b) Average delay τ (hours)

Figure 2.3: Performance Comparison of Independent and Botnet Methods in Spam distribution in Terms of Separate Objective.

Figs. 2.3 compare the two methods in terms of separate objectives including the number of lost legitimate followers and the average delay under different budget $|\mathcal{S}| = c$. We can see that both methods have the same coverage $|\mathcal{C}|$, which is equal to $|\mathcal{F}|$, as well as the maximum value of C. In addition, we can see from Fig. 2.3b that the delay of the independent method is zero, while that of botnet method could be on the order of hours. Finally, Fig. 2.3a shows that the botnet method has significant advantage than the independent method in terms of $|\tilde{\mathcal{S}}|$, the number of lost legitimate followers, as $|\tilde{\mathcal{S}}|$ is always equal to $|\mathcal{F}|$ for the independent scheme.

Finally, the botnet size n also has some impact on separate objectives in the botnet case. Fig. 2.3a shows that $\frac{|\tilde{\mathcal{S}}|}{|\mathcal{C}|}$ decreases as n increases. The reason is that the larger n, the more bots with less non-bot followers will be assigned to the first M levels, resulting in smaller $|\tilde{\mathcal{S}}|$ and thus larger $\frac{|\tilde{\mathcal{S}}|}{|\mathcal{C}|}$. In addition, Fig. 2.3b shows that the larger n, the higher the average delay τ, which is also expected.

In summary, from the view point of the botmaster, these evaluations show that the botnet scheme is more flexible than the independent method when considering multiple objectives of the spam distribution at the same time.

2.4 Social Botnet for Digital-influence Manipulation

In this section, we first briefly introduce digital influence and then experimentally show the efficacy of using the social botnet to manipulate digital influence.

2.4.1 Rise of Digital Influence

Digital influence is one of the hottest trends in social media and is defined as "the ability to cause effect, change behavior, and drive measurable outcomes online" in (122). The huge commercial potential of digital influence is in line with the increasingly recognized importance of word-of-mouth marketing on social networks. There

are also growing business cases in which various companies successfully promoted their services/products by reaching out to most influential social-network users in their respective context (122).

The future of digital influence also relies on effective tools to measure it. As reported in (122), there are over 20 popular digital-influence software vendors such as Klout (1) , Kred (2), Retweet Rank (3), PeerIndex (8), TwitterGrade (10) and Twitalyzer (9). Every vendor has its proprietary method to compute an *influence score* for a given user based on his activities within his affiliated social network such as Twitter, Facebook, Google+, and LinkedIn, and higher scores represent greater influence. As shown in Table 2.1, Klout, Kred, and PeerIndex use normalized scores with different average values and scales, while RetweetRank, TweetGrader, and Twitalyzer represent digital-influence scores using percentile.

The typical business model of digital-influence vendors is based around connecting businesses with individuals of high influence. Companies have paid to contact individuals with high influence scores in hopes that free merchandise and other perks will influence them to spread positive publicity for them. For example, in 2011 Chevy offered 139 3-day test drives of its 2012 Sonic model to selected participants with the Klout score of at least 35 (144). As another example, it has been reported that some recruiters have used the digital-influence scores to select qualified candidates (66). In addition, customer service providers like Genesys prioritize customer complaints according to their digital-influence scores to avoid the amplification of complaints by influential users in OSNs (75). Klout announced a growth of 2,000 new partners over a one year period in May 2012.

				Digital-influence score			
Vendor	Start	#users (M)	Update	Scale	Average	90-percentile	Target OSNs
Klout	2008	620+	daily	0-100	20	50	Twitter, Facebook, LinkedIn, Google+, Instagram, Foursquare
Kred	2011	-	hourly	0-1000	500	656	Twitter, LinkedIn, Facebook, Quora
PeerIndex	2009	50+	daily	0-100	19	42	Twitter, Facebook
Retweet Rank	2008	3.5	hourly	0-100	50	90	Twitter
Tweet Grader	2010	10+	daily	0-100	50	90	Twitter
Twitalyzer	2009	1	daily	0-100	50	90	Twitter

Table 2.1: Six Popular Digital-influence Software Vendors. The Data Was Collected at October, 2014.

2.4.2 *Botnet-based Digital-influence Manipulation*

Given the great potential of digital influence, whether it can be maliciously manipulated is an important research issue. For example, assume that malicious users could collude to significantly increase their influence scores. A company using the digital-influence service may consider them most influential and choose them as the targets of important marketing campaigns by mistake, thus having potentially huge financial loss, while malicious users can potentially benefit, e.g., by getting free sample products. In addition, malicious users may attract more legitimate followers who tend to follow most influential users and thus become more influential.

As the first work of its kind, we now explore the feasibility of using the botnet to manipulate digital influence. Our studies involve three most popular digital-influence vendors for Twitter users: Klout, Kred, and Retweet Rank. For clarity, we summarize their key features as follows.

- Klout: The Klout score of a Twitter user is on the scale of 1 to 100 and updated daily based on how frequently he or she is retweeted and mentioned in the last 90 days. The average Klout score is close to 20, and the score of the 90th percentile is over 50. [6]

- Kred: The Kred score of a Twitter user is on the scale of 1 to 1,000 and updated in real time according to how frequently he or she is retweeted, replied, mentioned, and followed on Twitter in the last 1,000 days. [7]

- Retweet Rank: It ranks the users based on how many times they each have been retweeted recently and how many followers/friends they each have. [8] Retweet

[6] `http://therealtimereport.com/2012/04/11/how-good-is-your-klout-score/`

[7] `http://kred.com/rules`

[8] http://www.retweetrank.com/view/about

ranks are updated on an hourly basis, and a retweet rank of x means that the corresponding user is the xth most influential on Twitter. A retweet rank score can also be translated into a percentile number ranging from 1 to 100, indicating how the user score comparing with other Twitter users.

Given a social botnet of n bots, we want to investigate whether it is feasible to generate an arbitrary influence score d_i for every bot $i \in [1, n]$ under each of the above three tools. Since every bot is usually indistinguishable from a legitimate user, our investigation can also shed light on the feasibility of using the botnet to manipulate the influence score of an arbitrary Twitter user. Since every digital-influence vendor (including the above three) usually keeps confidential its detailed algorithm for computing influence scores, our studies are purely based on real Twitter experiments. According to (122), we conjecture that the following three factors play the important role in determining a user's digital-influence score.

- *Actions.* Both the number and the type of actions have large impacts on a user's digital-influence score. Intuitively, the more actions the user can attract, the higher his digital-influence score. Moreover, different types of actions may have different impacts. For example, retweeting or replying should be more indicative than following because the latter has been shown to be more vulnerable to fake (151).

- *Audiences.* Given a target user u, we define all the users who have retweeted or replied u's tweets, or mentioned or followed u as u's *audiences*. We conjecture that the larger the audience size, the higher the digital-influence scores. The intuition is that the target user is more influential if each of his 100 tweets is retweeted by a different user than all 100 tweets are retweeted by the same one

user. We also conjecture that the higher the digital-influence scores his audience have, the higher the target user's digital-influence score.

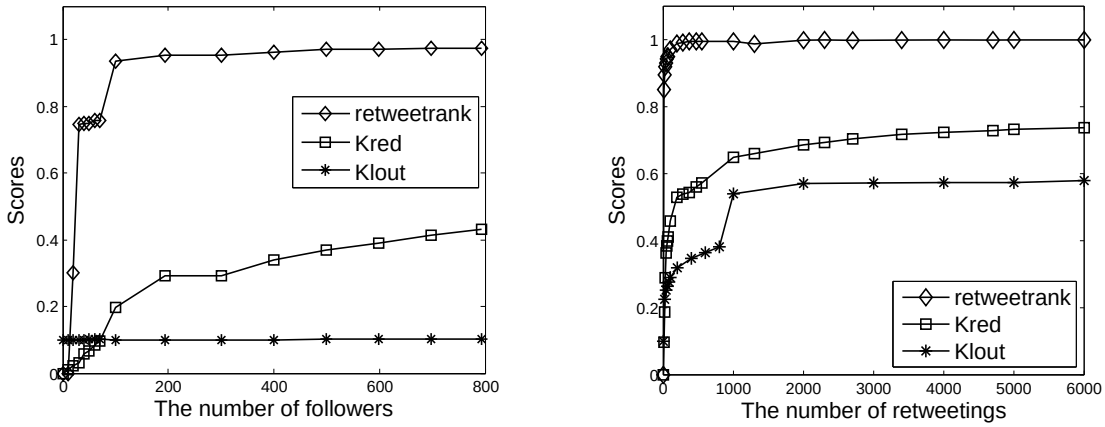
- *Popularity of tweets.* The digital-influence score of a target user is determined by the popularity of his tweets. We want to explore how the distribution of tweets popularity determine the target user's overall influence.

Based on these factors, we then present how to orchestrate the social botnets to manipulate the digital-influence scores.

Impact of Different Actions

Since almost all digital-influence tools measure a Twitter users digital influence as his ability to drive others to actions, our first experiment aims to evaluate the social botnet's impact with following, retweeting, and mentioning. We do not consider replying, as replying is treated by Twitter as a special type of mentioning and is thus expected to have the same effect as mentioning actions.

The first set of experiments involves $n = 1,000$ social bots, each of which has no interaction with any other Twitter account and hence is not scored by Klout, Kred, or Retweet Rank. Note that Klout assigns an influence score of 10 to new users, while Kred and Retweet Rank both assign a zero score to new users. We randomly choose three disjoint groups, each containing 10 social bots and performing a unique action. For every social bot in the *following* group, we add 10 randomly chosen social bots as followers each day of the first 10 days and then 100 followers each day of the next 10 days. Likewise, every social bot in the *retweeting* (or *mentioning*) group is retweeted (or mentioned) by randomly chosen social bots 10, 100 and 1000 times each day in the first, second, and the last 10 days, respectively. Since different vendors have different schedules for score updating, we report the social bots' influence scores observed at



(a) Impact of Following

(b) Impact of Retweeting

Figure 2.4: Manipulating Digital Influence by Following and Retweeting.

every midnight. In addition, since the three vendors have different score scales, we normalize different influence scores with respect to the corresponding maximum scores to facilitate direct comparison. In particular, we show $x/100$ and $y/1000$ for a Klout score x and a Kred score y, respectively, and report the percentile score for Tweet Rank.

Figs. 2.4a$\sim$2.4b show the impact of following and retweeting [9] actions on Klout, Kred, and Retweet Rank influence scores, where every data point is the average across the same group. We can see from Fig. 2.4a that the Klout influence score is not affected by the number of followers, while both Kred and Retweet Rank influence scores increase as the number of followers increases. This indicates that a social botnet can easily boost the Kred and Retweet Rank influence scores of its members by purposely following each other. Moreover, we can see from Fig. 2.4b that all three types of influence scores increase as the number that a user is retweeted increases.

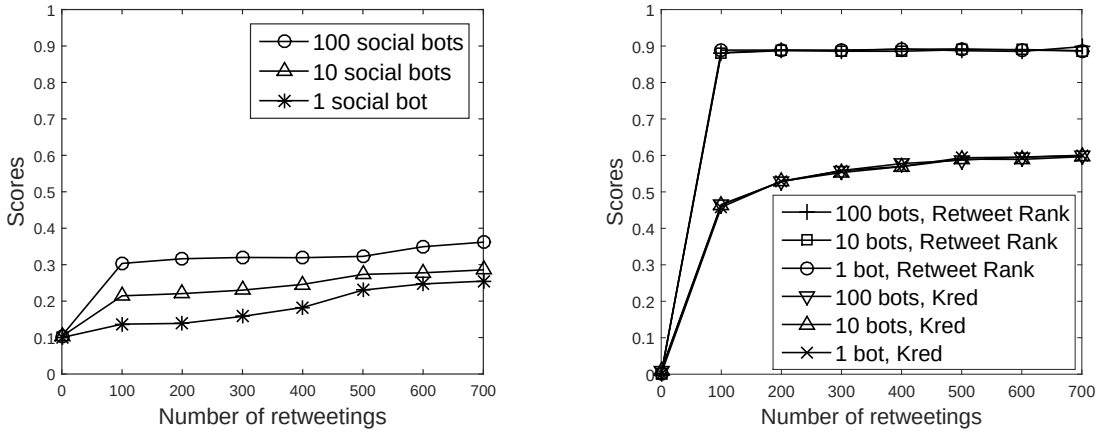
[9]Mentioning action has similar result with retweeting (161). We omitted here due to space constraints.

On the one hand, this makes much sense, as the higher the frequency in which a user is retweeted, the higher influence of that user has in its local neighborhood. On the other hand, this also renders the influence score measurement system vulnerable to social botnets, as colluding social bots can fake arbitrarily high retweeting frequency for any target user.

We can also see from Figs. 2.4a and 2.4b that none of our experiments has been able to escalate a user's influence score to an extremely high value, and we conjecture that there are two reasons for such difficulty. First, the audience size is limited, as we only have 1,000 social bots for experiment. We will show in the next set of experiments that the audience size has a large impact on the digital-influence scores. Second, it is likely that all the vendors have set up rules such that it is extremely difficult to achieve almost full digital-influence scores (1; 2; 3). Nevertheless, at the end of the experiments, Table 2.1 shows that all the digital-influence scores being manipulated have exceeded the 90-th percentile.

Impact of Different Audience Sizes

In this set of experiments, we measure the impact of different audience sizes on digital-influence scores. Recall that we define a user u's *audiences* as the set of users who have retweeted, mentioned, or followed u. We then try to answer the following questions. First, for two users with different audience sizes but the same numbers of actions, will the one with more audiences have a higher digital-influence score than the other? Second, it there an upper limit for a single social bot to manipulate the influence score of a target user? There could be two alternative answers to each of the two questions. On the one hand, if the digital-influence scores are related to both the number of incoming actions and the audience size, a single user should have limited power to manipulate the target user's influence score, and we thus need a large social



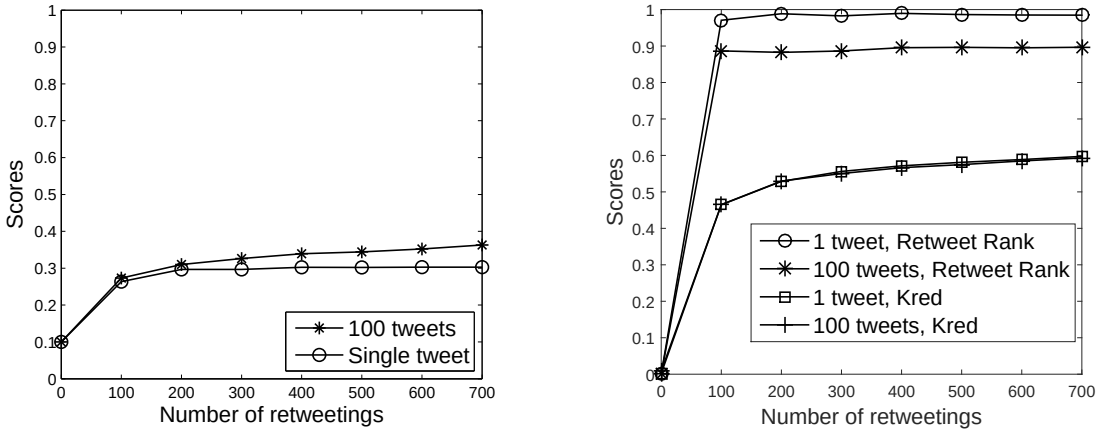
(a) Klout Scores

(b) Kred and Retweet Rank Scores

Figure 2.5: Manipulation by Social Botnets with Different Audience Sizes.

botnet to manipulate the target user's influence score to some extremely high value. On the other hand, if the digital-influence scores may not relate to the audience size, then 100 incoming actions from 100 different social bots would yield the same result as 100 incoming actions from a single social bot.

To verify which of the two conjectures is correct, we build three social botnets with 1, 10, and 100 bots, respectively. For each social botnet, we set 10 target users and retweet 100 tweets of each target each day for seven days. In other words, each bot in the three social botnets retweets 100, 10, and 1 times per day, respectively. Fig 2.5 shows the digital-influence scores of the 30 targets. As we can see, the audience size has no impact on both Kred and Retweet Rank scores but has large impact on the Klout scores. Specifically, the larger the audience size, the higher the Klout scores, and vice versa. Moreover, the Klout scores experience a sharp increase in the first day and then increase much slower in the following days. As a result, a single social bot can manipulate a target user's Kred and Retweet Rank scores with a large number



(a) Klout Scores

(b) Kred and Retweet Rank Scores

Figure 2.6: Manipulation by Acting on Different Number of Tweets.

of actions, while both large audience sizes and significant actions are necessary to obtain high Klout scores. We also conclude that Klout is more resilient to the social botnet than Kred and Retweet Rank.

Impact of Tweet Popularity

From the first set of experiments, we can see that the retweeting is the most effective way to manipulate a target user's digital-influence score. Given the same audience, the attacker can either retweet a single tweet of the target user to make this tweet very popular or retweet many of target user tweets so that each tweet will be less popular. We would like to answer which strategy will yield higher digital-influence score, i.e., whether tweet popularity has any impact on the digital-influence scores? To answer this question, we build a social botnet with 100 bots. We then select two groups with each containing 10 target users for manipulation. For the first group, each target first publishes a tweet, which is then retweeted by all the 100 social bots; while for the second group, each target user publishes 100 tweets, each of which is retweeted by

one social bot. We repeat this process daily and measure the digital-influence scores of the 20 targets. As we can see from Fig 2.6, the tweet popularity has no impact on both Klout and Kred scores and limited impact on the Retweet Rank score. In particular, adopting the second strategy will lead to a slightly higher Retweet Rank score of the target user.

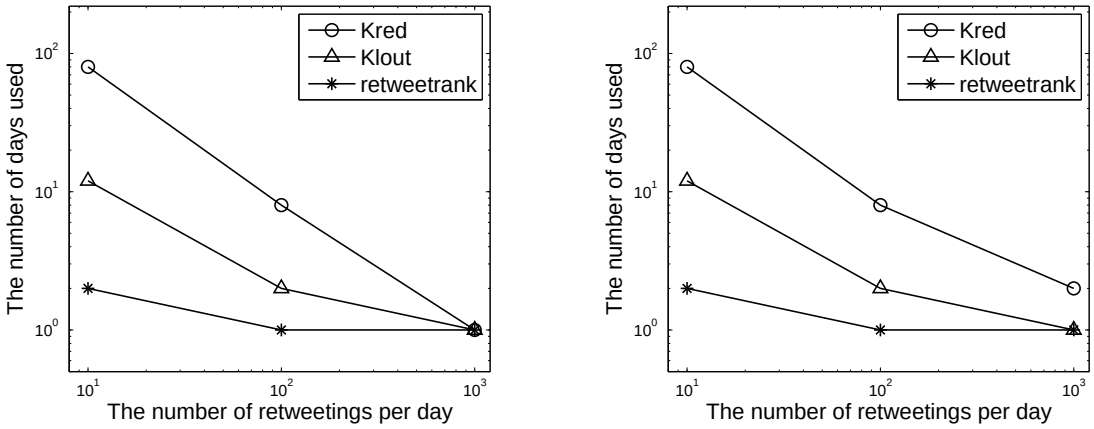
Speed of Digital-influence Manipulation

Our last experiment targets evaluating how fast influence scores can be manipulated. Same with the last set of experiments, we choose the retweeting as the action due to its effectiveness. For this purpose, we randomly select another three different groups of social bots, each containing 10 target bots. Every bot in the first, second, and third groups is retweeted 10, 100, and 1,000 times every day by random bots until the scores reach the 90th percentile, which corresponds to 50, 656, and 90 in Klout, Kred, and Retweet Rank, respectively.

Fig. 2.7 further shows the impact of retweeting frequency on the change of influence scores, where every data point represents the average across the same bot group. In particular, we can see that the number of days needed to increase the group average influence score from the initial value to 80-th or 90-th percentile is approximately inversely proportional to the retweeting frequency. In addition, we can see that for all three vendors, it is possible by retweeting 1000 times per day to reach the 80-th percentile and the 90-th percentile with only one and two days, respectively.

Remarks

We have two remarks to make. First, at the end of the experiments, no account was suspended by Twitter, as none of the accounts had conducted illegitimate activities such as spamming and aggressively following and unfollowing which could trigger sus-



(a) 80-th Percentile

(b) 90-th Percentile

Figure 2.7: Under Different Retweeting Speeds, the Number of Days Needed to Manipulate Digital Influence Scores from Nothing into 80-th and 90-th Percentiles.

pension by Twitter. In addition, both the spam distribution and the digital influence manipulation attacks show that social botnet has significant advantage over isolated bots. In particular, even though the attacker can disseminate spams to the same legitimate recipients using isolated bots, those isolated bots will be quickly suspended under Twitter's current policy or its variation and thus incurs significant cost for the attacker. Moreover, it is extremely difficult for the attacker to boost a single bot's digital influence score solely by letting the target bot tweet or retweet other legitimate users' tweets, as long as the digital influence is not entirely determined by outgoing interactions, as it is difficult for bots to attract incoming interactions from legitimate users.

2.5 Defenses

In this section, we propose two countermeasures for the two attacks above, respectively.

2.5.1 Defense Against Botnet-based Spam Distribution

Recall that in social botnet-based spam distribution, the attacker exploits retweeting trees to distribute spams (i.e., tweets with malicious URLs) such that only the bots on the first M (e.g., $M = 1$ in Twitter currently) levels of retweeting trees will be suspended.

To defend against this attack, we propose to track each user's history of participating in spam distribution and suspend a user if his accumulated suspicious behaviors exceed some threshold. Specifically, for each user v we maintain a *spam score* s_v, which is updated every time user v retweets a spam. Once s_v exceeds a predefined threshold, user v is labeled as a spammer and suspended.

We now discuss how s_v is updated. Our intuition is that the closer the user to the spam source, the more likely he is a member of the social botnet. The reason is that social botnet usually prefers shorter retweeting path for fast spam dissemination, while a spam tweet traversing a long retweeting path, i.e., sequentially retweeted by many accounts, incurs large delay and gives more time for Twitter to detect them. To reflect this idea, whenever a user retweets a spam, we update his spam score as

$$s_v = s_v + \gamma^d \,, \tag{2.2}$$

where d is the number of retweeting hops between the spam source to v, and $\gamma \leq 1$ is the attenuation factor of distance. Note that $d = 0$ if user v is the source of a spam. In this way, any user who has retweeted spams is punished by having his spam score increased. Once a user's spam score exceeds certain predetermined threshold, the user is suspended. Note that Twitter currently suspends a user whenever he has published one spam tweet. To mimic the current Twitter policy, we can set the threshold as one.

Evaluation

Similar to Section 2.3.3, we built a Twitter subnet composed of 6000 legitimate users and 400 social bots. Each social bot has $|\mathcal{F}_i|$ legitimate followers, where $|\mathcal{F}_i|$ is drawn from Gaussian distribution with $\mu = 32$ and $\delta^2 = 5$, and each social bot follows all other social bots. We assume that the attacker builds the optimal retweeting tree according to Section 2.3.2. We adopt similar parameters as in Section 2.3.3 by setting $M = 3$, $\alpha = 0.2$, $c = 10$, and $K = 10$.

To model the spam campaign, we conduct the experiment in multiple rounds. In each round, we randomly select one social bot to publish a spam tweet and other social bots that have not been suspended to retweet the spam tweet according to the retweeting forest. We assume that the legitimate users retweet the spam occasionally with probability β. At the end of each round, we update the spam scores for each bot and legitimate user according to Eq. (2.2). If $s_v \geq 1$ for some user v, we remove v from the simulated network. We then start the next round by randomly selecting a bot as the new source and continue the spam distribution. To reduce the randomness, we repeat each experiment 100 times and report the averages.

We evaluate the proposed defense in comparison with three other baseline defenses as follows.

- **Defense I**. The original defense that suspends the users in the first M levels from the spam source, which is considered by the attacker to launch the optimal spam distribution as discussed in Section 3.2.

- **Defense II**. Any user who retweets δ spams is suspended, regardless of its distance to the spam source, where δ is a system parameter.

- **Defense III**. Assume that user v has retweeted a spam tweet t within M hops from the source and that t has been retweeted by n_t users in total. The spam

score for v is updated as

$$s_v = s_v + 1/\log(1 + n_t) \ .$$

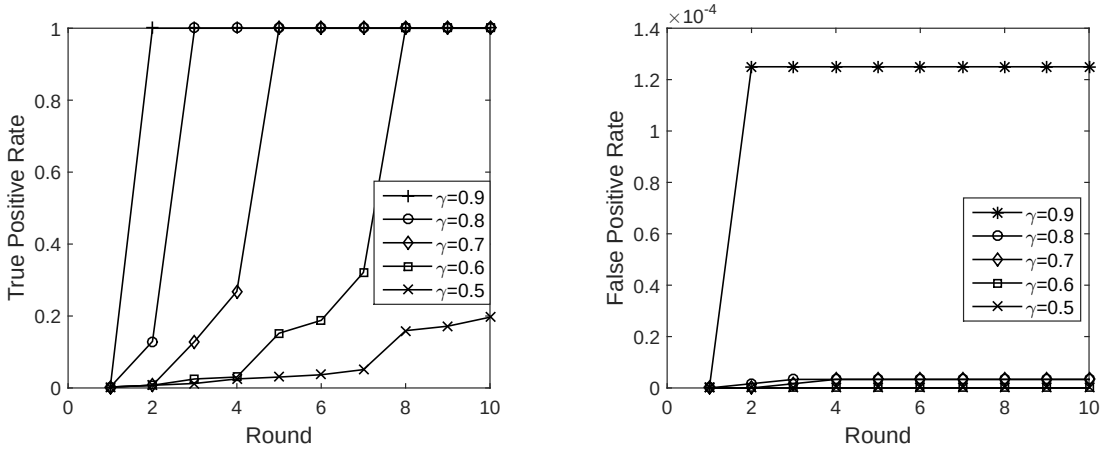
Defense III extends Defense I and II by taking the popularity of individual spam tweet into account. The intuition is that the more users retweet a spam tweet, the more deceiving the spam tweet, and the less likely that any individual user who retweets it is a bot. We also use the system parameter δ as the suspending threshold for this defense scheme.

- **Defense IV**. The proposed defense.

We use four metrics to compare the proposed defense with the three baseline defenses. To begin with, let N_{bots} and N_{legit} be the numbers of social bots and legitimate users, respectively. Also let S_{bots} and S_{legit} be the numbers of suspended social bots and legitimate users, respectively. We define the following four metrics.

- **True positive rate (`TPR`):** the ratio of the suspended bots over the total bots, which can be computed as $S_{\text{bots}}/N_{\text{bots}}$. This metric is also referred to as recall.
- **False positive rate (`FPR`):** the ratio of suspended legitimate users over all the legitimate users, which can be computed as $S_{\text{legit}}/N_{\text{legit}}$.
- **Precision:** the ratio of suspended bots over all suspended users, which can be computed as $S_{\text{bots}}/(S_{\text{legit}} + S_{\text{bots}})$.
- **Overall performance:** $\texttt{TPR} - P \cdot \texttt{FPR}$ where P is the penalty parameter for `FPR`.

We adopted overall performance here because `FPR` has much larger impact than `TPR` from the service provider's point of view, as suspending a legitimate user damages its reputation and incurs much severer consequences than a social bot evading detection.



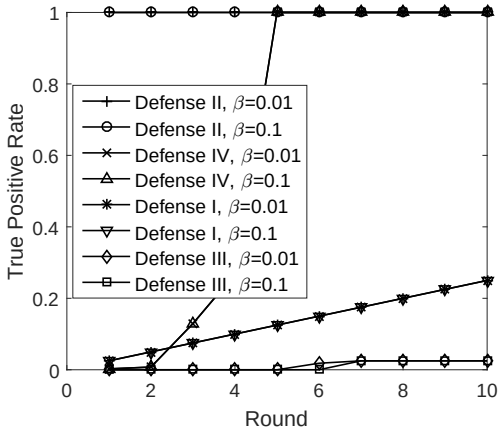
(a) True Positive Rate

(b) False Positive Rate

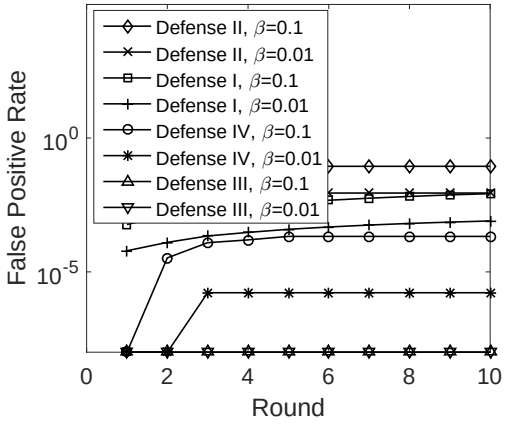
Figure 2.8: The True and False Positive Rates with Different γs.

Fig. 2.8 shows the true positive and false positive rates with different γs which are the attenuation factors of distance. As we can see, the proposed defense could quickly detect all the social bots when γ is large. Specifically, when $\gamma = 0.9$, all the social bots can be detected and suspended in the second round. Moreover, as expected, the ratio of true suspension at the same round will decrease as γ decreases. Finally, there is an anticipated tradeoff between false and true positive rates for different γs. The larger γ, the higher true positive rate but also the higher false positive rate, and vice versa. In the following experiments, we set $\gamma = 0.7$ by default.

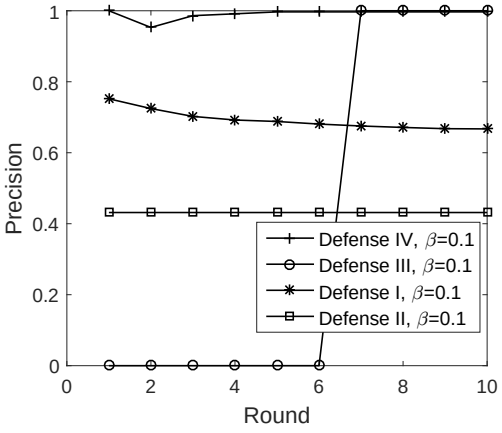
Fig. 2.9 compares the proposed defense with the three baselines under different βs, i.e., the probability of a legitimate user retweeting a spam tweet. We can make five observations here. First, Fig. 2.9a shows that β has no impact on the `TPR`, but Fig. 2.9b shows that the larger β, the higher `FPR` for all four defenses, which is consistent with the definition of β. Second, Fig. 2.9a shows that **Defense II** has the highest `TPR` = 100% in every round. The reason is that when $\delta = 1.0$ all the social bots will be suspended after the first round. **Defense IV** has the second highest



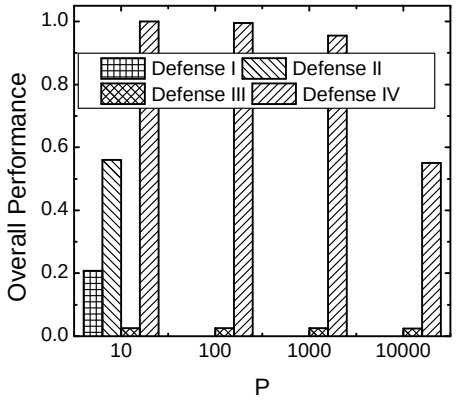
(a) True Positive Rate (Recall)



(b) False Positive Rate



(c) Precision



(d) Overall Performance

Figure 2.9: Performance Comparison.

TPR and needs about five rounds to approach 100%. **Defense I** and **III** both have lower TPR because they only examine the users in the first $M = 3$ levels instead of the whole network. Third, Fig. 2.9b shows that the FPR of the proposed defense is lower than those of **Defenses I** and **II** but slightly higher than that of **Defense III**. This is because **Defense I** and **II** have much less restrict conditions for suspending users than **Defense IV**. Fourth, Fig. 2.9c shows that **Defenses IV** and **III** have the precision close to 100% after the first and sixth rounds, respectively, as they both incur low false positives and sufficiently high true positives. **Defense III** has 0% of precision before the sixth round because it did not suspend any social bot in the first five rounds. In contrast, both **Defense I** and **II** have relatively low precision because they misclassified many normal users as bots. Finally, Fig. 2.9d shows that the proposed defense has better overall performance than all three baselines with penalty factor P varying from 10 to 10000. The reason is that **Defense IV** has much lower FPR than **Defense I** and **II** and higher TPR than **Defense III**. In summary, the proposed defense outperforms all three baselines as it can effectively detect social bots while being friendly to legitimate users.

2.5.2 Defense Against Digital-influence Manipulation

As discussed in Section 2.4, all the digital-influence software vendors consider the number of actions and/or the audience size as the major factors in evaluating a user's digital-influence, which makes them vulnerable to the social botnet. To address this vulnerability, we propose a new digital-influence measurement scheme inspired by (63). The key idea is to find sufficient credible users and only use the actions from them to compute digital-influence scores for other users. Specifically, given a social network with user set V and all the actions (including following, retweeting, mentioning, and replying) among the users, we first find a subset $V^* \subseteq V$ of users

that are credible.

We then define the digital-influence score of each user v based on the actions from the credible user set V^*. Specifically, assume that user v has received a_j actions from each user $j \in V^*$. The digital influence score for v is then given by as

$$\mathsf{score}(v) = \sum_{j \in V^*} f(a_j) \tag{2.3}$$

where

$$f(a_j) = \begin{cases} a_j & \text{if } a_j = 0, 1, \\ 1 + \lambda \cdot \exp(-\frac{1}{a_j}) & \text{else,} \end{cases} \tag{2.4}$$

and λ is a system parameter that represents the maximum impact of actions from a single user on one's score. In practice, we can set $\lambda = 1$ such that a single user could contribute the score by at most 2. It is easy to see that the above score definition takes both the number of actions and the audience size into account, which captures the key ideas behind Klout, Kred, Retweet Rank scores as well our findings in Section 2.4.2.

The challenge is then how to find the credible user set V^*. To tackle this challenge, we observe that although the actions among the social bots and from the social bots to the legitimate users are unpredictable, legitimate users usually carefully choose whom to interact, so there are much fewer actions from legitimate users to social bots. Based on this observation, we first find a small number of trusted users, which could either be the verified users maintained by Twitter or manually verified.

We then assign each trusted user some initial credits and distribute the credits along the action edges in multiple rounds. In each round, every user with credits keeps one unit of credit for himself and distributes the remaining credits to his neighbors along the action edges. The credit distribution process terminates when no user has extra credits to distribute. Since legitimate users are more likely to receive credits than social bots during the process, every user who has received at least one unit of

credit is considered credible.

More specifically, we first construct an action graph for credit distribution. Given the user set V and all their actions during the period T, we build a weighted and directed action graph $\mathcal{G} = (V, E)$, where each user corresponds to a vertex in V, and there is an arc e_{ij} from user i to user j an edge with the weight w_{ij} if user i has retweeted, replied, or mentioned user j for w_{ij} times during the period T. We do not consider the following action because it has been reported that normal users could carelessly follow back whoever has followed them (151; 59).

We then design a credit distribution scheme on $\mathcal{G} = (V, E)$, which consists of seeds selection, initial credit assignment, iterative credit distribution, and termination. To initiate credit distribution, we first select a seed set S from V which are trusted users such as the verified users maintained by Twitter or manually verified, and partition the whole graph into a tree of multiple levels, in which the seed users occupy the first level, their one-hop outgoing neighbors occupy the second level, and so on. A node is assigned to the highest level if its incoming neighbors appear at different levels. We then assign each seed s with the amount of credits proportional to the sum of weights of its outgoing edges, i.e., $w_o(s)C_{\text{total}}/\sum_{s\in S} w_o(s)$, where C_{total} is the total amount of initial credits, and $w_o(s)$ is the sum of weights of the seed s's all outgoing edges. We then distribute the credit from the seed users along the tree level by level. Specifically, if a user v at the nth level has $c(v)$ units of credit, he holds one unit of credit for himself and distributes the remaining $c(v) - 1$ units of credit to its outgoing neighbors at the $(n + 1)$th level, where the amount of credits received by neighbor v' is proportional to the edge weight $e_{vv'}$. In other words, neighbor v' receives $w_{vv'}(c(v)-1)/\sum_{u\in\mathcal{O}(v)w_{vu}}$ units of credits. The credits could only be distributed from one level to the immediate higher level and are rounded to the closest integer. The credit distribution terminates if none of the nodes has any extra credit to distribute

to its outgoing neighbors.

The choice of C_{total} represents the tradeoff between true and false positive rates. On the one hand, with a larger C_{total}, we could discover more credible users and compute digital-influence scores for more legitimate users at the cost of more social bots obtaining credits and being labelled as credible users. On the other hand, a smaller C_{total} will result in fewer credible users being discovered as well as fewer social bots being labelled as credible users. As in (135), we set $C_{\text{total}} = O(\sqrt{|V|})$ to strike a good balance between true and false positive rates. The idea is that at the end of credit distribution, approximately C_{total} users will each hold one unit of credit and be labelled as credible. If we set $C_{\text{total}} = O(\sqrt{|V|})$ and the interaction graph is well-connected, with high probability there exists at least one arc from some credible users to each of the $|V|$ users while the credit distribution will terminate in $O(\log |V|)$ levels with decreasing credits per arc such that each attack arc from legitimate users to social bots is expected to receive only $O(1)$ of credits (135).

Evaluation

We follow the approach taken by (155; 29; 160) to evaluate the performance of the digital-influence measurement scheme. Specifically, we first use the Twitter geo-search API (13) to collect the users in a specific metropolitan area and then crawl the latest 600 tweets of each user to extract the interactions such as retweets, replies, and mentions. We create four datasets from the four representative area in U.S. as shown in Table 2.2.

For each dataset, we build one ground-truth network composed of both legitimate users and social botnets. Specifically, we first build one *legitimate action subgraph* from the interactions among the legitimate users in the that area, where each vertex corresponds with one legitimate user, and an edge from one user to another corre-

Table 2.2: Four Action Networks for Evaluation, Where 'F' and 'I' Refer to Following and Interaction, Respectively.

Area	#Users	#F-edges	#I-edges (#Weights)
Tucson (TS)	28,161	830,926	162,333 (669,006)
Philadelphia (PI)	144,033	5,462,013	1,434,375 (4,972,689)
Chicago (CI)	318,632	14,737,526	3,631,469 (12,010,986)
Los Angeles (LA)	300,148	18,333,774	4,391,542 (14,048,838)

sponds to interaction from the former to the latter. We further create a completely-connected social botnet with the same number of social bots as the legitimate users. We then construct a ground-truth network by connecting the legitimate action subgraph and the social botnet with a number of *attack edges* from legitimate action subgraph to the social botnet. Here we assume that there is no edge from the social botnet to the legitimate action subgraph, which is clearly in favor of the social bots, as the social bots would only lose credits through such edges.

The attack edges are constructed in the following way. Assuming that there are total g edges in the legitimate action subgraph, we create ωg attack edges with unit weight, where ω is the ratio representing *attack strength*. As in (160), we vary the ω from 10^{-5} to 10^{-4} of the total edges in each legitimate subgraph. We use ground-truth networks to conduct the credit distribution and find the credible user subset V^*.

As in (29), we consider two types of attacks. In the *random attack*, we randomly choose ωg legitimate users and connect each of them to one randomly chosen social bot with an attack edge. In this case, the attacker is unaware of which users are seed. In the *seed-targeting attack*, we create ωg attack edges between the social botnet and ωg users randomly chosen from $100\omega g$ legitimate users with shortest distances to the seeds. This type of attack mimics the case that the attacker tries to acquire positions

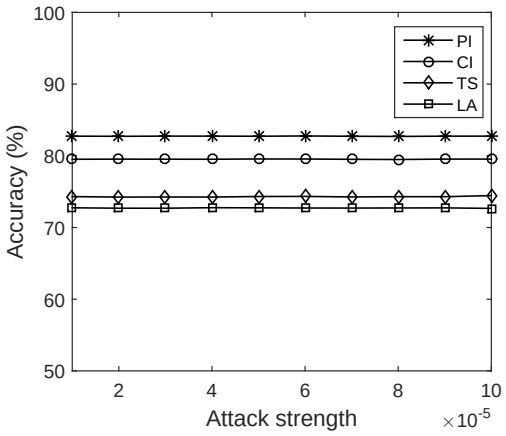
close to the seed users to gain more credits during credit distribution. For both attacks, we assume that the social botnet can arbitrarily distribute received credits internally. For example, suppose that the social botnet receives C_{bots} credits after the credit distribution, they distribute these credits to C_{bots} social bots each with one credit, such that all C_{bots} bots become credible users.

We evaluate the performance of the proposed digital-influence measurement scheme using the following metrics.

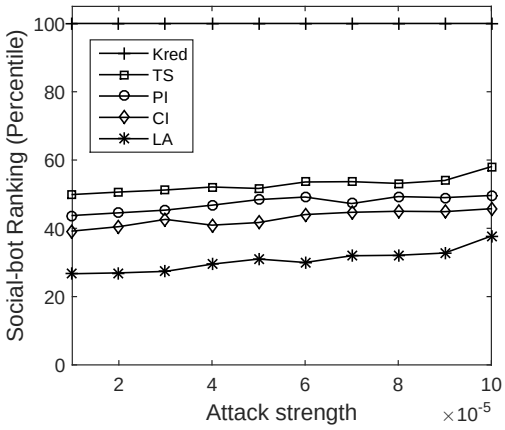
- *Top-K-percent accuracy.* Let $\mathcal{U}_1$ be the list of legitimate users ranked by their total numbers of incoming interactions. Also let $\mathcal{U}_2$ be the list of legitimate users ranked by their digital influence scores obtained by applying the proposed digital-influence measurement scheme to the ground-truth network. Further let $\mathcal{U}_1(K)$ and $\mathcal{U}_2(K)$ be the top-K-percent users in $\mathcal{U}_1$ and $\mathcal{U}_2$, respectively. The top-K-percent accuracy is defined as $\frac{|\mathcal{U}_1(K) \cap \mathcal{U}_2(K)|}{|\mathcal{U}_1(K)|}$. The more accurate of the proposed scheme, the higher the ratio of common users in these two top-K lists, and vice versa.

- *Social-bot influence ranking.* Let bot b be the bot with the highest digital influence score output by the proposed digital-influence measurement scheme. Social-bot influence ranking is defined as b's rank in percentile in $\mathcal{U}_1$. The lower the bot's percentile, the higher resilience to the social botnet, and vice versa.

Fig. 2.10 and Fig. 2.11 show the *top-10-percent accuracy* and the *social-bot influence ranking* under two types of attacks with attack strength varying from 10^{-5} to 10^{-4}. In general, we see similar trend in all scenarios for all the four datasets, meaning that the results are consistent across different datasets. Moreover, in all cases, the accuracy is always larger than 70%, meaning that the proposed digital-influence measurement scheme can discover 70% of top-10% influential users under the social

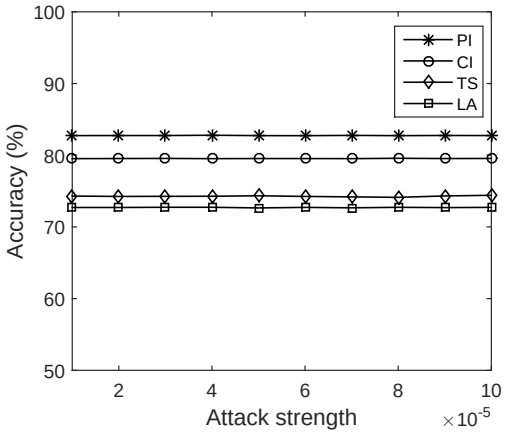


(a) Top-10-percent Accuracy

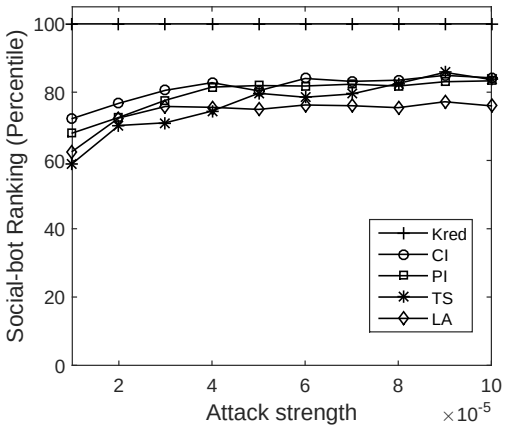


(b) Social-bot Influence Ranking

Figure 2.10: The Performance under the Random Attack.



(a) Top-10-percent Accuracy



(b) Social-bot Influence Ranking

Figure 2.11: The Performance under the Seed-targeting Attack.

botnet attack. We can also see in Fig. 2.10a and Fig. 2.11a that the the accuracy is insensitive to the change of attack strength. The reason is that the top influential users attract actions from many legitimate users, who have higher chances to acquire the credits and become credible users in the presence of attack edges. In contrast, as shown in in Fig. 2.10b and Fig. 2.11b, the social-bot influence ranking increases as the attack strength increases under both types of attacks. This is expected, because as attack strength increases, more social bots will acquire credits to become credible users and affect target bots' influence scores. We can also see from Figs. 2.10b and 2.11b that the social-bot influence ranking is below top 40% and top 20% under the random attack and seed-targeting attack, respectively.

Fig. 2.10b and Fig. 2.11b also compare the proposed defense with the existing influence measurement vendor Kred. [10] As we can see, the bot's influence score could always rank at the first position because we assumed that there are infinite actions between any two bots in the botnet. In contrast, the proposed scheme could effectively defend against the manipulation from social botnets.

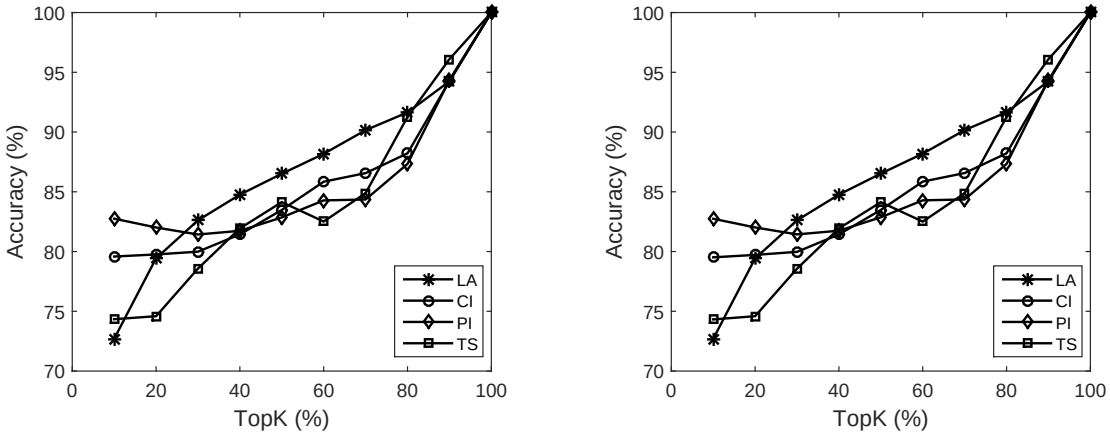
Fig. 2.12 shows the impact of K on the top-K-percent accuracy, which generally increases until to 100% when K increases from 10 to 100. Overall, the accuracy is higher than 70% in the figures, and $K = 10$ in previous experiments is representative among the worst cases.

In summary, experiment results show that the proposed digital-influence measurement scheme is resilient to the social botnet attack.

2.6 Related Work

In this section, we discuss the prior work tightly related to our work in this chapter.

[10] we choose Kred because only Kred has publish its influence score model on `http://kred.com/rules`.



(a) Random Attack

(b) Seed-targeting Attack

Figure 2.12: The Impact of K on the Top-K-percent Accuracy.

The social botnet has received attention only recently. Ferrara *et al.* summarized the research on social bots in (52). Boshmaf *et al.* showed that a social botnet is very effective in connecting to many random or targeted Facebook users (i.e., large-scale infiltration) (26). The work in (152; 53) shows how the spammers become smarter to embed themselves into Twitter. Messias *et al.* used two sicalbots to manipulate the Klout and Twitalyzer scores by following and tweeting actions(101). Our preliminary work (161) was independently done from (101) and shows how social bots could cooperate to manipulate the influence scores of Klout, Kred, and Retweet Rank.

There is a rich literature on spam detection in OSNs. The first line of work such as (157; 123; 131; 56; 125; 82; 150; 20; 141; 83) considers independent spam bots and comes up with different methods to characterize and detect them. For example, Song *et al.* found that the distance between the sender and receiver of a message is larger for spammers than legitimate users (123). Zhang *et al.* found that automated spammers usually show some temporal pattern which can be used for detection (157). Another features and the corresponding machine learning effort

can be found in (131; 56; 125; 82; 150; 20; 141; 83). Some of these results such as (123) have been incorporated into our design constraints in § 2.3.2.

The second line of work such as (57; 62; 132; 35; 151; 59) focuses on characterizing and detecting organized spam campaigns launched by an army of spam bots. We discover a new method for effective spam campaign in this chapter, and whether the results in (57; 62; 132; 35; 151; 59) can be directly or adapted to detect our method is certainly challenging and worthy of further investigation. Moreover, spam bots are evolving towards more intelligence. Yang *et al.* found that instead of separate communities, spam bots embedded themselves into the network by building small-world-like networks between them and also maintaining tight links with external legitimate users(151). Meanwhile, Ghosh *et al.* discovered the similarly rich connections between spam bots and the legitimate users(59). Our work is consistent with this trending and explore the new attacks by using the spam bots' growing intelligence.

There are effective solutions such as (29; 138; 139; 135; 154; 155) to detecting Sybil accounts in distributed systems under the control of a single attacker. These solutions commonly assume that the link between two accounts corresponds to a real social relationship difficult to establish and impossible to forge. Moreover, all of these system assume that the connection is undirected. In this chapter, we designed the defense scheme for directed and weighted Twitter network by using the similar observation that the amount of interactions from a legitimate user to a social bot is usually far less than that in the reverse direction in § 2.5.2. We also found that using the interaction network yields better defense performance than the following networks. Although § 2.5.2 has illustrate an example by using the trustworthiness of interactions to defend against the specific digital-influence manipulation attack, we can use the same observation but different methods to identify the social botnets and

then eliminate them from the microblogging system.

Also related is the research on computing the digital influence score (30; 143; 16). For example, Cha *et al.*, compared three influence ranking lists in Twitter by counting the number of retweets, mentions and followers and found that the influenced defined by the number of followers is very different with the other two metrics (30). Bakshy *et al.* (16) proposed to measure user influence based on his ability to post the tweets that generates a cascade of retweets. This line of research does not consider the impact of the social botnet. This work is also different from our previous work (160) which focuses on finding the sybil-resilient top-K influential users in Twitter.

2.7 Summary

In this chapter, we firmly validated the efficacy and merits of botnet-based spam distribution and digital-influence manipulation on Twitter through thorough experiments and trace-driven simulations. We also propose the countermeasures corresponding to these two attacks and demonstrate their effectiveness.

Chapter 3

TRUETOP: A SYBIL-RESILIENT SYSTEM FOR USER INFLUENCE MEASUREMENT ON TWITTER

3.1 Introduction

Influential Twitter users have great potential for accelerating information dissemination and acquisition. For example, to launch a viral marketing campaign for a new product via Twitter, a known strategy is for the marketer to seed the product with a few selected influential users who can potentially influence a disproportionately large number of others and also quickly trigger a cascade of influence. As another example, in the event of a national crisis, the governmental authority can conduct a massive information campaign by disseminating truthful information via influential users to effectively achieve strategic goals and also counteract rumors. As the last example, to have realtime situational awareness about a physical region of interest, military agencies can recruit volunteers in the target region via influential Twitter users there and then outsource the collection of in-situ information to the volunteers.

The strong promise of influential users leads to the growing attention on how to measure the influence of a Twitter user (30; 78; 143; 16). There are also over 20 commercial tools available for measuring twitterers' online influence. Common to these research proposals (30; 78; 143; 16) and commercial tools is to capture the qualitative feature of online influence as "the ability to cause effect, change behavior, and drive measurable outcomes online" (122) and to quantify a twitterer's online influence based on his/her *interactions* with others.

The rise of *social bot*s (52) or *sybil*s (45) in general on Twitter is jeopardizing trustworthy influence measurement. In a sybil attack, the adversary coordinates many fake accounts (also called *bot*s or *sybil user*s hereafter) to unfairly overpower non-sybil users. Despite various efforts to detect sybil users on Twitter (20; 62; 125; 131; 56; 133), sybil users are still thriving on Twitter. For example, a recent study (4) revealed that at least 10% of Twitter users are sybil users. Given the exclusive reliance of existing influence measurement techniques on user interactions, the adversary could coordinate his sybil users to create arbitrary interactions to inflate their influence scores on Twitter. Since influence scores are relatively defined, the adversary could also effectively deflate the influence scores of non-sybil Twitter users. According to our recent study (161), an adversary controlling 1,000 sybil users can quickly generate an influence score in the 95th percentile for any sybil user under popular influence measurement tools such as Klout (1), Kred (2), and Retweet Rank (3). In a similar study (101), Messias *et al.* used two social bots to successfully obtain high Klout scores.

The lack of sybil-resilient influence measurement services on Twitter can be detrimental. Specifically, there is a growing market for influence measurement services with more than 20 service providers available (122). If these service providers fail to provide trustworthy measurement results due to sybil attacks, they will have extreme difficulty getting customers and surviving, and their customers could not achieve effective information dissemination or acquisition as expected.

The root cause for the vulnerability of existing influence measurement techniques to sybil attacks lies in the incautious use of user interactions. Specifically, Twitter permits four types of publicly visible user interactions, including *follow*, *retweet*, *reply*, and *mention*. The interactions about any user can be further classified into *incoming* interactions towards him and *outgoing* interactions from him. Since a sybil user can

freely follow, retweet, reply to, and mention other sybil or non-sybil users, extensive outgoing interactions are fairly easy to create and thus unsuitable for sybil-resilient influence measurement. In addition, since sybil users could easily get many legitimate followers (59; 151; 126), the number of followers each user has should also be ruled out. In contrast, we observe from real Twitter data that non-sybil users tend to be more selective in retweeting, replying to, and mentioning other users. This observation is in line with the real-life scenario: one may exchange business cards with many strangers but will be more cautious in choosing whom to further interact with. This means that incoming retweets, replies, and mentions are much more trustworthy information for measuring user influence. Existing influence measurement techniques, however, use all incoming and outgoing interactions in a non-discriminative way.

We propose TrueTop, a novel sybil-resilient influence measurement system based on the incoming retweets, replies, and mentions each Twitter user has. TrueTop provides on-demand influence measurement services to various customers such as business companies and government agencies. Given a target set of Twitter users (e.g., those in a geographic area of interest), TrueTop outputs a ranked list of top-K influential users for a desirable integer $K \geq 1$. TrueTop is designed to be *sybil-resilient* and also *accurate*, which means that the TrueTop output contains bounded sybil users and the true top-K non-sybil users with overwhelming probability, respectively.

The main design challenge for TrueTop is that sybil users can arbitrarily interact among themselves, so it is not sybil-resilient to evaluate a user's influence directly based on his total incoming retweets, replies, and mentions. We propose the following method to tackle this challenge. Given the target set of users, we first construct a weighted directed *interaction graph*, in which every vertex corresponds to a unique user in the target set. An edge from vertex a to vertex b exists if user a has ever retweeted, replied to, or mentioned user b, and the edge weight is proportional to the

number of retweets, replies, and mentions from a to b. Imagine that the interaction graph consists of a virtual non-sybil region with all non-sybil users and a virtual sybil region with all sybil users. Given our previous observations, both the number of edges and the total edge weights from the non-sybil region to the sybil region should be much smaller than those in the reverse direction. Then we seed some carefully chosen vertices (or users) in the non-sybil region with some *credits* and let every vertex in the whole graph allocate its current credits to its direct successors proportionally to the corresponding edge weights in every iteration. After sufficient iterations, the top-K influential non-sybil users are very likely to stand out, as they can accumulate many credits due to their abundant incoming retweets, replies, and mentions. In contrast, the total credits flowing into the sybil region can be very limited, so even the sybil users with many incoming interactions from sybil followers may end up with few credits. We can thus achieve sybil-resilient influence measurement by counting the final credits at every vertex.

This chapter makes the following contributions.

- We motivate and formulate the problem of sybil-resilient influence measurements on Twitter.
- We propose TrueTop, a novel influence measurement system that can identify the top-K influential users in a target set of Twitter users with high accuracy in the presence of sybil attacks by exploiting the selectivity of non-sybil users in interacting with other users.
- We confirm the high accuracy and sybil-resilience of TrueTop by detailed theoretical analysis and extensive experiments on real datasets.

The rest of this chapter is organized as follows. Section 3.2 surveys the related work. Section 5.3.5 introduces Twitter basics, our system and threat models, and our

design objectives. Section 3.4 illustrates the TrueTop design. Section 3.5 theoretically analyzes the accuracy and sybil resilience of TrueTop. Section 3.6 evaluates the performance of TrueTop by detailed experiments. Section 3.7 summarizes the chapter.

3.2 Related Work

There is significant effort to explore social networks for effective sybil defenses in various distributed systems, such as SybilGuard (155) and SybilLimit (154) for P2P networks, SumUp (135) for online voting systems, and SybilInfer (41), SybilDefense (142), and SybilRank (29) for online social networks. A common assumption is that each node can be mapped into one in an undirected social network graph where every edge corresponds to a human-established trust relation. Although the attacker can create many sybil accounts, he cannot establish an arbitrarily large number of social trust relations with non-sybil users. Moreover, all schemes assume that the honest region is fast mixing and separate from the sybil region. Built upon these two key insights, these schemes conduct varying community detection methods (140) to limit the number of sybil users admitted into or their impact in various application scenarios.

Recent measurement studies have questioned these two assumptions. Yang *et al.* (153) showed that sybil users on the Facebook-like Renren network can have their friend requests accepted by many non-sybil users. A similar result targeting Facebook was reported in (26). Blending sybil users into the non-sybil community would reduce the effectiveness of the existing sybil defenses (76). In addition, the work in (120; 59; 101; 161; 52) showed that sybil users successfully acquired a number of followings from non-sybil users on Twitter. All these findings indicate that neither bidirectional friendships in Fackbook-like OSNs nor unidirectional followings in Twitter-like microblogging systems can be used as the trustable mirroring of real social relations.

Moreover, it has been shown in (105; 104) that the mixing time of many practical and directed social graphs is much longer than previously expected. Since neither of the two key assumptions underlying the schemes in (155; 135; 41; 140; 154; 142; 29) holds in directed networks such as Twitter, they are not directly applicable to our targeted scenario. Our TrueTop system does not rely on either assumption.

As a special kind of sybil users, spammers in Twitter has attracted considerable attention in recent years. A common approach adopted by existing work (20; 125; 62; 131; 56; 133; 49; 68) is to detect spammers by measuring the behavioral difference between spammers and legitimate users. Spammers are a special type of sybil users, and the detection of general sybil users on Twitter remains an open challenge.

There is a rich literature for influence measurement on Twitter. Cha *et al.* (30) found that the numbers of retweets and mentions serve as better metrics than the number of followers in measuring user influence. Bakshy *et al.* (16) proposed to measure user influence based on his ability to post the tweets that generates a cascade of retweets. TwitterRank (143) combines link structure and topical similarity between Twitter users and uses a modified PageRank algorithm to calculate user influence. Pal and Counts (114) also proposed a framework to identify topical authorities in microblogging systems. All these schemes are vulnerable to sybil users who can forge arbitrary information employed by these schemes for influence measurement. Moreover, many metrics used by these schemes have been incorporated into commercial influence measurement tools (122), and the vulnerability of representative tools to sybil attacks has been experimentally verified in (161).

Also related is the research on modelling, measuring, and analyzing the interactions in OSNs, e.g., (36; 21; 145; 71; 147). Our work is the first to build a weighted directed interaction graph from historical incoming retweets, replies, and mentions on Twitter and use it for identifying influential users.

3.3 Preliminaries

3.3.1 Twitter Basics

We illustrate the basic operations on Twitter to help understand our design. The social relationships on Twitter are unidirectional by users *following* others. If user A follows user B, A is B's *follower*, and B is A's *followee*. A user usually needs no prior consent from his followees. Twitter also allows each user to approve/deny every following request, but this option is relatively rarely used. A user can send text-based messages of up to 140 characters, known as *tweet*s, which can be read by all his followers. Tweets can be visible to anyone with or without a Twitter account, and they can also be protected and are only visible to approved followers. There are three special kinds of tweets corresponding to three operations. A *retweet* is a re-posting of someone else's tweet, a *reply* corresponds to a response to a tweet, and a *mention* refers to inserting "@username" in a tweet to ensure that the specified user can see this tweet. Finally, each user has a *timeline* which shows all the latest tweets (including original tweets, retweets, replies, and mentions) of his followees. Also note that Twitter allows direct messages to be sent between users. Since those direct messages are not publicly visible, they cannot be used to measure user influence.

3.3.2 System Model

TrueTop is run by a service provider (SP) offering on-demand influence measurement services to customers such as viral marketers, government/military agencies, or even individuals. Given a measurement request, the TrueTop SP first determines the target set of Twitter users to evaluate, denoted by $\mathcal{U}$. The users in $\mathcal{U}$ can be directly given by the customer or identified by the TrueTop SP according to some common features specified by the customer. For example, the customer can specify a target

geographic region, a target age group, a target topic (e.g., music), etc. As said, TrueTop relies on incoming interactions among the users in $\mathcal{U}$, i.e., the retweets, replies, and mentions each user in $\mathcal{U}$ has received from all the other users in $\mathcal{U}$. So we assume that the SP has a reliable way to obtain the incoming interaction data needed, e.g., directly from Twitter, via crawling, or from some third-party providers of social media data. For example, Gnip (`http://gnip.com/`) is an authorized reseller of Twitter data. TrueTop is designed to output a ranked list of top-K influential users in $\mathcal{U}$, where $K \geq 1$ denotes a customer-specified integer.

3.3.3 Threat Model

Let $\tilde{\mathcal{U}}$ denote all possible sybil users in $\mathcal{U}$. We assume that the SP knows neither which user in $\mathcal{U}$ is a sybil user nor how many sybil users there are; otherwise, the identified sybil users can be simply removed from $\mathcal{U}$. Based on the recent measurement study (59), we assume that each sybil user may have followed and also been followed by some non-sybil and sybil users in $\mathcal{U}$. There may be a single attacker controlling $\tilde{\mathcal{U}}$ or multiple independent ones with each controlling an exclusive subset of $\tilde{\mathcal{U}}$. TrueTop can deal with both cases without modification, so we focus on the more challenging former case hereafter. The goal of the attacker is to gain high influence scores for his sybil users and maximize the number of users in the TrueTop output.

3.3.4 Design Objectives

Let $\mathcal{U}_K^*$ and $\mathcal{U}_K$ denote the top-K non-sybil influential users in $\mathcal{U}$ and the TrueTop output, respectively. We have two major design objectives.

- *Accuracy*: TrueTop should identify the true top-K non-sybil users, which means the difference between $\mathcal{U}_K^*$ and $\mathcal{U}_K$ should be very small.

- *Sybil resilience*: TrueTop should not identify sybil users as top-K users, i.e., the the intersection $\mathcal{U}_K \cap \tilde{\mathcal{U}}$ should be very small.

3.4 TrueTop Design

3.4.1 Overview

TrueTop is motivated by the observation that incoming retweets, replies, and mentions are more trustworthy for measuring user influence than outgoing interactions. So our first step is to construct an interaction graph, in which every vertex corresponds to a unique user in the target set $\mathcal{U}$, and every directed edge indicates totally non-zero retweets, replies, and mentions from the tail user to the head user. In addition, the weight of every edge is a non-decreasing function of related retweets, replies, and mentions.

The next step is to choose a suitable metric to quantify the influence of every user (vertex) in the interaction graph. TrueTop adopts *weighted eigenvector centrality* (WEC for short) (108), the de facto metric for measuring the influence of a node in a weighted directed graph. Specifically, the WEC score of every user corresponds to his influence score, which depends on the weights of his incoming edges, the number of his direct predecessors, and their influence scores which are further determined by their respective incoming edges and direct predecessors. The WEC score reflects an intuition that the influence of a user is better indicated by the interactions from influential users than those from less influential users.

We uses iterative credit distribution for the convenience to describe and understand our method. Specifically, we select some random users (called *seed*s) in the interaction graph and seed each with some *credit*s. In each iteration, we allocate all the credits each user receives in the last iteration to his direct successors propor-

tionally to individual edge weights. The credits each user receives in one iteration are expected to stabilize after sufficient iterations and be proportional to his WEC score. It can be easily shown that iterative credit distribution is equivalent to power iteration (80), a standard technique for computing WEC scores. Since sybil users can create arbitrary interactions among themselves, some of them may gain enough credits to appear in the top-K list. TrueTop achieves high sybil resilience by carefully choosing the initial seeds and also early terminating iterative credit distribution.

In what follows, we first illustrate the construction of the interaction graph in Section 3.4.2. Next, we present an iterative credit distribution scheme over the interaction graph in Section 3.4.3. Finally, we introduce how to achieve sybil-resilient iterative credit distribution in Section 3.4.4.

3.4.2 Interaction Graph Construction

Given the target users $\mathcal{U}$ and their interaction data, TrueTop first builds a weighted directed interaction graph denoted by $\mathcal{G} = \langle \mathcal{U}, \mathcal{V} \rangle$, where $\mathcal{U}$ is abused to denote the vertex set, and every edge $v_{i,j} \in \mathcal{V}$ $(i, j \in \mathcal{U})$ is directed and indicates that there are some retweets, replies, and/or mentions from user i to j. The major challenge here is to determine the weight $w_{i,j}$ of every edge $v_{i,j}$. As shown in Fig. 3.1, $\mathcal{G}$ can be divided into a virtual sybil region $\mathcal{S}$ including all the sybil users and a virtual non-sybil region $\mathcal{H}$ including all the non-sybil users. The sybil-resilience requirement for TrueTop requires that the sum of the edge weights from the non-sybil region to the sybil region is small, while the accuracy requirement for TrueTop demands that the weight $w_{i,j}$ reflects the true influence of user j on i in the target period. Let $\mathcal{I}_{i,j}$ denote the set of time-indexed retweets, replies, and mentions from user i to j. We consider the following two methods for defining the edge weights.

- *Sum-based.* In this method, $w_{i,j}$ equals $|\mathcal{I}_{i,j}|$. Sum-based edge weights satisfy

the sybil-resilient requirement, as the total edge weights from the non-sybil region to the sybil region are as limited as the number of retweets, replies, and mentions from non-sybil users to sybil users. They also partially satisfy the accuracy requirement, as the more interactions from i to j, the more influence j likely has on i, and the higher $w_{i,j}$. Sum-based edge weights, however, fail to catch the temporal aspect of interactions. For example, consider another direct predecessor of j, say l, where $|\mathcal{I}_{i,j}| = |\mathcal{I}_{l,j}|$. Assume that the interactions in $\mathcal{I}_{l,j}$ occurred in the last few days in the target period, while those in $\mathcal{I}_{i,j}$ were spread more evenly. It may be natural to say that j has stronger influence on user i than on user l, but we have $w_{i,j} = w_{l,j}$ for sum-based methods.

- *Entropy-based.* In this method, we divide the target period into μ equal-length epochs for some system parameter $\mu \geq 1$ and denote the total number of retweets, replies, and mentions from user i to j in epoch xth by d_x, where $|\mathcal{I}_{i,j}| = \sum_{x=1}^{\mu} d_x$. Then we define the edge weight $w_{i,j} = (1 - \sum_{x=1}^{\mu} \frac{d_x}{|\mathcal{I}_{i,j}|} \log \frac{d_x}{|\mathcal{I}_{i,j}|})|\mathcal{I}_{i,j}|$. The more consistent the interactions from i to j in time, the higher $w_{i,j}$, and vice versa. When all the interactions happen in a single epoch, the weight is identical to sum-based $|\mathcal{I}_{i,j}|$. Entropy-based edge weights can also satisfy the sybil-resilience requirement, as non-sybil users unlikely have consistent interactions to sybil users so that the total edge weight from the non-sybil region to the sybil region can be expectedly small. In contrast to sum-based edge weights, entropy-based edge weights successfully catch the temporal information in the interactions while failing to reflect the volume of the interactions. So they partially satisfy the accuracy requirement as well.

The effects of the above methods are compared in Section 3.6. There may be other ways to define the edge weights. For example, we can let $w_{i,j}$ equal a linear combina-

tion of the edge weights derived under sum-based and entropy methods, respectively; we can also assign different weights to retweets, replies, and mentions according to slightly different effort and/or social implication related to performing these interactions. A further study on such issues is left as future work due to space constraints.

Note that we only consider retweets, replies, and mentions in the weight definitions because they are representative on Twitter and have been used in all the existing influence measurement techniques. Some other factors could also impact the user influence, such as following connections and favorites. As stated before, since sybil users could easily get many legitimate followers (59; 151; 126), the following connections fail to achieve the sybil resilience and hence should be ruled out for the influence measurement. On Twitter, a user could favor the tweets from other users, but there is no public Twitter API which can return the favorite user list for any given tweet. Should a public Twitter API for retrieving favorites become available, we can easily incorporate favorites into TrueTop.

3.4.3 Credit Distribution

TrueTop uses the WEC score of every user in $\mathcal{G} = \langle \mathcal{U}, \mathcal{V} \rangle$ as his influence score. Specifically, let π_i denote the WEC score of user i in $\mathcal{G}$ and $\mathbf{W} = (w_{i,j})$ denote the normalized weighted adjacency matrix of $\mathcal{G}$. The vector $\boldsymbol{\pi} = \langle \pi_1, \pi_2, \ldots, \pi_{|\mathcal{U}|} \rangle$ is the dominant eigenvector of $\mathbf{W}$, i.e., the solution to the equation $\boldsymbol{\pi}\mathbf{W} = \boldsymbol{\pi}$ according to (108).

Power iteration (80) is a common technique to compute the WEC vector $\boldsymbol{\pi}$. Let $\mathbf{v}_0$ be a random vector composed of $|\mathcal{U}|$ nonnegative elements totalling one. In power iteration, $\boldsymbol{\pi}$ is computed in an iterative fashion as

$$\boldsymbol{\pi} = \lim_{t \to \infty} \mathbf{x}^{(t)} = \lim_{t \to \infty} \mathbf{v}_0 \mathbf{W}^t \ , \tag{3.1}$$

where $\mathbf{x}^{(t)} = \mathbf{x}^{(t-1)}\mathbf{W}$ with the initial $\mathbf{x}^{(0)} = \mathbf{v}_0$. If $\mathcal{G}$ is strongly connected, $\boldsymbol{\pi}$ exists, is unique, and is unrelated to $\mathbf{v}_0$. In practice, power iteration normally terminates if $\|\mathbf{x}^{(t)} - \mathbf{x}^{(t-1)}\|_1 < \nu$ for some acceptable error threshold ν (e.g., 10^{-9}).

The WEC vector only exists in a strongly connected graph (108), in which every vertex is reachable from every other vertex. Although $\mathcal{G}$ itself may be not strongly connected in practice, it usually has a giant strongly connected component (GSCC) which includes the majority of the vertexes and edges and is dramatically larger than all other strongly connected components (SCCs). Since the most influential users should have intensive interactions with other users, the top-K influential users should be in the GSCC with overwhelming probability. Our subsequent operations thus apply to the GSCC only. The verification of the existence of GSCC in real datasets is deferred to Section 3.6.

TrueTop uses iterative credit distribution instead to compute $\boldsymbol{\pi}$ to facilitate the presentation. Initially, we randomly select a few users (called *seed*s) in $\mathcal{G}$ and initialize each with the same number of notional credits totalling one. At every iteration, we allocate the credits each user receives in the last iteration to his direct successors proportionally to the corresponding edge weights. Let $C_j^{(t)}$ denote the number of credits at any user $j \in \mathcal{U}$ after t iterations, which are proportional to his influence score measured after t iterations. $C_j^{(t)}$ is a real number in general and can be computed as

$$C_j^{(t)} = \sum_{i \in \texttt{IN}(j)} \frac{w_{i,j} C_i^{(t-1)}}{\sum_{k \in \texttt{OUT}(i)} w_{i,k}}, \tag{3.2}$$

where $\texttt{IN}(j)$ and $\texttt{OUT}(i)$ denote the direct predecessors of user j and the direct successors of user i in $\mathcal{G}$, respectively. Similarly, we can terminate credit distribution when $\sum_{j \in \mathcal{U}} |C_j^{(t)} - C_j^{(t-1)}| < \eta$ for some acceptable error threshold η (e.g., 10^{-9}).

We can easily show that iterative credit distribution above is equivalent to power

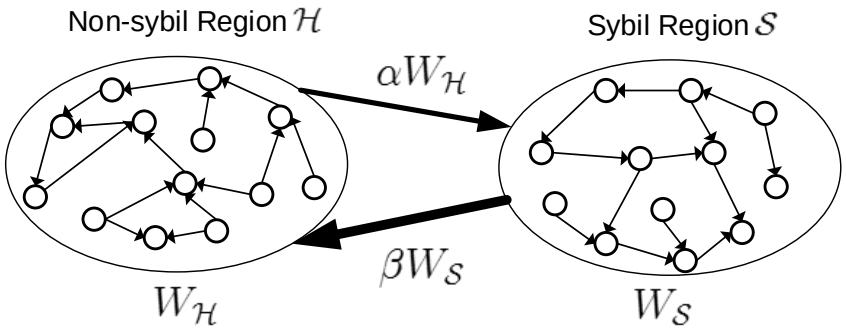


Figure 3.1: The Interaction Graph With a Virtual Non-sybil Region $\mathcal{H}$ and a Virtual Sybil Region $\mathcal{S}$.

iteration. In particular, assume that s seeds are chosen in iterative credit distribution, each having $1/s$ credits initially. We further select $\mathbf{v}_0$ for power iteration such that the ith element equals $1/s$ if user i is a seed and zero otherwise. Then Eq. (3.2) is apparently the element-wise expression of $\mathbf{x}^{(t)} = \mathbf{x}^{(t-1)}\mathbf{W}$. Since power iteration does not depend on a specific $\mathbf{v}_0$, we have $x_j^{(t)} = C_j^{(t)}$ for any user $j \in \mathcal{U}$ after t iterations.

Iterative credit distribution described above is still subject to sybil attacks. To see this, consider Fig. 3.1 where the interaction graph is divided into a virtual non-sybil region $\mathcal{H}$ and a virtual sybil region $\mathcal{S}$. We denote the total edge weights within $\mathcal{H}$, within $\mathcal{S}$, from $\mathcal{H}$ to $\mathcal{S}$, and from $\mathcal{S}$ to $\mathcal{H}$ by $W_{\mathcal{H}}$, $W_{\mathcal{S}}$, $\alpha W_{\mathcal{H}}$, and $\beta W_{\mathcal{S}}$, respectively, where $\alpha \ll 1$. Although the adversary has no control over $W_{\mathcal{H}}$ and α, he can easily manipulate $W_{\mathcal{S}}$ and β to make $\beta W_{\mathcal{S}}$ very small. Even if all the seeds are chosen from $\mathcal{H}$ in the best scenario, more and more credits will flow into and stay in $\mathcal{S}$ as time goes by. We have the following proposition about the vulnerability of iterative credit distribution to sybil attacks.

Proposition 3.4.1. *Assume that the total edge weights from the non-sybil region $\mathcal{H}$ to the sybil region $\mathcal{S}$ and from $\mathcal{S}$ to $\mathcal{H}$ are α and β fractions of the total edge weights in*

$\mathcal{H}$ and $\mathcal{S}$, respectively. The total credits in $\mathcal{S}$ increase monotonically with the iteration t and asymptotically approach to $\frac{\alpha}{\alpha+\beta}$.

Proof. Let the total credits in $\mathcal{H}$ and $\mathcal{S}$ at t-th iteration be $C_{\mathcal{H}}^{(t)}$ and $C_{\mathcal{S}}^{(t)}$, respectively. According to the credit distribution defined in Eq. 3.2, after the t-th iteration, the average credits flowed from $\mathcal{H}$ to $\mathcal{S}$ and $\mathcal{S}$ to $\mathcal{H}$ are $\alpha C_{\mathcal{H}}^{(t)}$ and $\beta C_{\mathcal{S}}^{(t)}$, respectively. Meanwhile, the total credits in the whole network is constant to 1. Hence,

$$\begin{aligned}
C_{\mathcal{H}}^{(t)} &= (1-\alpha)C_{\mathcal{H}}^{(t-1)} + \beta C_{\mathcal{S}}^{(t-1)} \\
&= (1-\alpha)C_{\mathcal{H}}^{(t-1)} + \beta(1 - C_{\mathcal{H}}^{(t-1)}) \\
&= (1-\alpha-\beta)C_{\mathcal{H}}^{(t-1)} + \beta \\
&= (1-\alpha-\beta)^{t-1}C_{\mathcal{H}}^{(1)} + ((1-\alpha-\beta)^{t-2} + \ldots + 1)\beta \\
&= (\frac{1}{\alpha+\beta} - 1)\alpha(1-\alpha-\beta)^{t-1} + \frac{\beta}{\alpha+\beta}
\end{aligned}$$

and

$$C_{\mathcal{S}}^{(t)} = 1 - C_{\mathcal{H}}^{(t)} = (1 - \frac{1}{\alpha+\beta})\alpha(1-\alpha-\beta)^{t-1} + \frac{\alpha}{\alpha+\beta}$$

Since $\alpha \ll 1$ and $\beta \ll 1$, $C_{\mathcal{H}}^{(t)}$ will decrease monotonically and $C_{\mathcal{S}}^{(t)}$ will increase monotonically. When $t \to \infty$, $C_{\mathcal{S}}^{(t)} = \frac{\alpha}{\alpha+\beta}$.

□

Since the adversary can well control the topology within $\mathcal{S}$, most credits in $\mathcal{S}$ can go to a few sybil users who may eventually appear in the top-K influential users.

3.4.4 Sybil-Resilient Credit Distribution

TrueTop adopts the following two defenses against sybil attacks such that most credits can stay in the non-sybil region for sufficient iterations.

The first defense is to use non-sybil seeds only so that credit distribution can start from the non-sybil region $\mathcal{H}$. We propose to use verified Twitter users as seeds

by three reasons. First, Twitter has certified their authenticity. Each verified user has a blue verified badge on his profile page and is followed by the official Twitter account *@verified*. Second, there are many verified users available as candidate seeds. As of April 2014, Twitter has verified more than 88,600 accounts among 255 million monthly active users and keeps verifying more. Since $\mathcal{G}$ can be expected to contain many users in practice, there should be at least one verified user in $\mathcal{G}$ with very high probability. Finally, since verified users are usually public figures such as politicians, celebrities, or business leaders, we can trust them to be very cautious in whom to retweet, reply to, and mention. This implies that the immediate successors of verified users on the interaction graph $\mathcal{G}$ are very likely to be non-sybil users as well, so are the successors' immediate successors. If we start credit distribution from verified users, most credits can be expected to stay inside $\mathcal{H}$ after many iterations.

How many seeds should we choose? Some verified users may be very close to the sybil region, but we cannot tell who they are. Ideally speaking, we should choose the verified users far from the sybil region. On the one hand, if a verified user is randomly chosen as the sole seed, he may be too close to the sybil region. On the other hand, if we use all the verified users in $\mathcal{G}$ as the seeds, it is very likely that some of them are close to the sybil region. In addition, the number of seeds affects the convergence of iterative credit distribution: the more seeds, the faster the algorithm converges. It is impossible to specify the decisive rules for seed selection, so we randomly choose $s \geq 1$ seeds from the verified users in $\mathcal{G}$ and experimentally evaluate the impact of seed selection in Section 3.6.

How should we assign the initial credits among the s seeds? We propose two methods as follows.

- *Basic method.* The total credits are evenly assigned to the s seeds. This straightforward method assumes that each seed has the same importance for credit

distribution.

- *Reverse-WEC.* Since the credits flow out from the seeds, we can assign more initial credits to the seeds who can quickly reach more users to speed up the algorithm convergence. For this purpose, we conduct the credit distribution introduced in Section 3.4.3 over an inverse interaction graph generated from $\mathcal{G}$ by reversing the directions of all the edges and also setting all the edge weights to one. The final credits at each user naturally reflects his connectivity in $\mathcal{G}$. So we select the verified users with the top-K highest credits as the seeds and then assign to each of them the initial credits proportional to their credits obtained via reverse credit distribution.

The second defense is to early terminate iterative credit distribution before it converges in the whole graph $\mathcal{G}$. To see the necessity and intuition for this defense, recall that we start credit distribution from non-sybil seeds in the non-sybil region. Since the total edge weight from the non-sybil region to the sybil region is relatively small, we can expect credit distribution to converge much faster in the non-sybil region than in the whole $\mathcal{G}$. In addition, the most influential non-sybil users normally have many incoming interactions and thus a rich number of credit sources in $\mathcal{G}$. So they can quickly accumulate a lot of credits to stand out much faster than other non-sybil users. If we early terminate iterative credit distribution, most or all of the sybil users would not get enough credits to appear in the resulting top-K influential users, so we can achieve sybil resilience. However, if credit distribution stops too early, some true top-K influential non-sybil users may not get enough credits to be ranked in the top-K list, leading to an inaccurate result.

We design a simple but effective algorithm to tackle the dilemma between sybil resilience and accuracy. The key idea is to monitor the ranking change of the can-

didate top-K users in two consecutive iterations. Whenever the ranking change is no larger than an acceptable threshold, we terminate the algorithm and output the current top-K users as the top-K influential users. This algorithm is directly built on our observation above. Specifically, since the top-K non-sybil influential users is more likely to stand out much faster than both sybil users and other non-sybil users during credit distribution, their rankings are more likely to become stable in fewer iterations as well. We detail the algorithm as follows and postpone its performance analysis to Section 3.5.

Algorithm 1: Find the top-K influential users

input : Interaction graph $\mathcal{G}$; s seed users; K; maximum number of iterations T; ranking-error tolerance ϵ

output: The top-K influential users

```
Assign initial credits among s seed users by either basic or reverse-WEC method;
t ← 1;
while t < T do
    Distribute the credit in the t-th iteration according to Eq. 3.2;
    Rank the users by their credits and obtain the candidate top-K users R^(t);
    Compute the ranking distance d(K)^(t) between R^(t) and R^(t−1) as in Eq. 3.3;
    if d(K)^(t) <= ε then
        break;
    t ← t + 1;
return R^(t) as the top-K influential users
```

Let $r^{(t)}(u)$ and $r^{(t-1)}(u)$ denote the rankings of user u in iterations t and $t-1$,

respectively. We define the ranking distance $d(K)^{(t)}$ between $\mathcal{R}^{(t)}$ and $\mathcal{R}^{(t-1)}$ as

$$d(K)^{(t)} = \sum_{u \in \mathcal{R}^{(t)}(K) \cup \mathcal{R}^{(t-1)}(K)} |r^{(t)}(u) - r^{(t-1)}(u)| \ . \tag{3.3}$$

The algorithm above has two key parameters: T and ϵ. The former dictates the maximum number of iterations, and the latter specifies the maximum ranking error tolerance. The larger T, the longer the algorithm execution time, the more accurate the top-K influential users, the more credits flowing into the sybil region and thus the less sybil resilience, and vice versa. In contrast, the larger ϵ, the shorter the algorithm execution time, the less accurate the top-K influential users, the fewer credits flowing into the sybil region and thus the higher sybil resilience, and vice versa. In practice, we can let $\epsilon < K$, meaning that each user in the current top-K list has experienced a ranking change of less than one on average in contrast to the previous iteration.

3.5 Performance Analysis

In this section, we analyze the accuracy and sybil resilience of TrueTop. Recall that $\mathcal{U}_K^*$ denotes the true top-K influential users in the non-sybil region, $\mathcal{U}_K$ denotes the TrueTop output (i.e., the output of Alg. 10), and $\tilde{\mathcal{U}}$ denotes all the sybil users in the sybil region. So we can use $\mathcal{U}_K \cap \mathcal{U}_K^*$ and $\mathcal{U}_K \cap \tilde{\mathcal{U}}$ to measure the accuracy and sybil-resilience of TrueTop, respectively.

To make the performance analysis tractable, we first assume that Alg. 10 runs in the non-sybil region only, so we can conduct an upper-bound analysis about the accuracy of TrueTop by setting the ranking error tolerance parameter $\epsilon = 0$ and T extremely large such that Alg. 10 terminates only when a stable top-K user list is found. We then show that Alg. 10 will terminate in asymptotically the same number of iterations for $\epsilon = 0$, based on which we finally estimate the number of sybil users appearing in $\mathcal{U}_K$. As stated before, the larger ϵ, the shorter the algorithm execution

time, the less accurate the top-K influential users, the fewer credits flowing into the sybil region and thus the higher sybil resilience, and vice versa. Hence by setting $\epsilon = 0$, we can provide the lower and upper bounds for sybil resilience and accuracy, respectively. As for arbitrary $\epsilon > 0$, we unfortunately cannot obtain the closed-form analytical result for sybil resilience or accuracy and thus resort to experiments to evaluate its impact in Section 3.6.

The following concepts are needed for the accuracy analysis.

Definition 3.5.1 ((Relative) Error Bound)**.** *Let $\boldsymbol{\pi}$ denote the true WEC vector of non-sybil users and the k-ranked user refer to the one with the kth highest WEC score τ_k in $\boldsymbol{\pi}$. Let $\tau_k^{(t)}$ denote the WEC score of the k-ranked user after iteration t. Then $e_k^{(t)} = |\tau_k^{(t)} - \tau_k|$ is defined as the error bound for the k-ranked node after iteration t, and $e_k'^{(t)} = e_k^{(t)}/\tau_k$ is defined as the relative error bound.*

Definition 3.5.2 ((Relative) WEC gap)**.** *The WEC gap for the k-ranked node is defined as $\Delta_k = \tau_k - \tau_{k+1}$, and $\Delta_k' = \Delta_k/\tau_k$ is the correspondingly relative WEC gap.*

Lemma 3.5.1. *Let $\mathbf{W}$ denote the normalized weighted adjacency matrix of the non-sybil region with n users, among which there are s seed users. Construct $\mathbf{v}_0$ for power iteration (see Eq. 3.1) such that the ith element equals $1/s$ if user i is a seed and zero otherwise. Then the relative error bound for the k-ranked user satisfies $e_k'^{(t)} \leq \lambda^t$, where $\lambda < 1$ denotes $\mathbf{W}$'s second largest eigenvalue.*

Proof. According to the Perron-Frobenius theory (18), the matrix $\mathbf{W}$ is irreducible and has the largest eigenvalue of 1, and all other eigenvalues are absolutely less than 1, denoted as $1 = \lambda_1 > \lambda_2 \geq \lambda_3 \geq \ldots \geq \lambda_n > -1$. Moreover, if we denote the corresponding n eigenvectors as $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$, then $|\mathbf{v}_1| = 1$ and we denote $\mathbf{v}_1$ as the WEC vector $\boldsymbol{\pi}$. Next if $\mathbf{W}$ is diagonalizable, then $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ can be orthogonal

to expand the whole space of $\mathbb{R}^n$. For the case of non-diagonalizable $\mathbf{W}$, we can use the Jordan canonical form to transform it into a diagonalizable one (121).

Since $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \ldots, \mathbf{v}_n$ are orthogonal, $\mathbf{v}_0$ can be written as

$$\mathbf{v}_0 = \sum_{i=1}^{n} a_i \mathbf{v}_i \tag{3.4}$$

where $a_i \in \mathbb{R}$. We argue that if $\mathbf{W}$ is stochastic and irreducible then $a_1 = 1$. To see why, we first notice that since $\mathbf{W}$ is stochastic, $\mathbf{W1} = \mathbf{1}$. It follows that $\mathbf{v}_i^T \mathbf{W1} = \lambda_i \mathbf{v}_i^T \mathbf{1} = \mathbf{v}_i^T \mathbf{1}$. The eigenvector corresponding to λ_1 is the stationary distribution of Markov Chain $\mathbf{W}$. Since $\mathbf{W}$ is irreducible, $\lambda_i < 1$ and $\lambda_i \neq 1$ when $i \neq 1$. Thus we can see that $\mathbf{v}_i^T \mathbf{1} = 0$ for $i \neq 1$. Multiplying $\mathbf{1}$ at both sides of Eq. 3.4, it follows that $\mathbf{v}_0 \mathbf{1} = a_1 \mathbf{v}_1 \mathbf{1}$. Since both $\mathbf{v}_0$ and $\mathbf{v}_1$ are non-negative vectors with the sum of 1, we have $a_1 = 1$.

Thus Eq. 3.4 can be simplified as

$$\mathbf{v}_0 = \mathbf{v}_1 + \sum_{i=2}^{n} a_i \mathbf{v}_i = \boldsymbol{\pi} + \sum_{i=2}^{n} a_i \mathbf{v}_i \ .$$

Multiplying $\mathbf{W}^t$ at both sides and keeping using the equation $\mathbf{v}_i \mathbf{W} = \lambda_i \mathbf{W}$, we can obtain

$$\mathbf{x}^{(t)} = \mathbf{v}_0 \mathbf{W}^t = (\boldsymbol{\pi} + \sum_{i=2}^{n} a_i \mathbf{v}_i)\mathbf{W}^t = \boldsymbol{\pi} + \sum_{i=2}^{n} \lambda_i^t a_i \mathbf{v}_i \ .$$

Let $\lambda = max(|\lambda_2|, |\lambda_n|)$. As the $t \to \infty$, λ^t will become dominant and it follows that $|(\mathbf{x}^{(t)} - \boldsymbol{\pi})_i| = O(\lambda^t)$

Moreover, for $j \in \mathcal{U} \setminus \mathbf{s}, \sum_{i=1}^{n} a_i v_{i,j} = v_{0,j} = 0$. Hence,

$$e_j^{(t)} = |\sum_{i=2}^{n} \lambda_i^t a_i v_{i,j}| \leq \lambda^t |\sum_{i=2}^{n} a_i v_{i,j}| = \lambda^t |- v_{1,j}| = \lambda^t \pi_j \ .$$

□

Lemma 3.5.1 states that the rank of each user in iteration t approaches its true rank for sufficiently large t.

In addition, Ghoshal and Barabasi (60) recently found that if the WEC vector (Pagerank in their paper) follows power law distribution, the gap between the kth and $(k+1)$th WEC scores decreases with k. We thus have the following lemma.

Lemma 3.5.2. *(60) If the WEC vector $\boldsymbol{\pi}$ follows a power-law distribution with parameter γ, the relative WEC gap for the k-ranked user satisfies $\Delta'_k \approx \frac{1}{k(\gamma-1)}$.*

The proof of Lemma 3.5.2 is straightforward according to (60) and omitted here due to space constraints. In Section 3.6, we show that the WEC vectors for real Twitter datasets indeed follow the power-law distribution. We then have the following theorem based on Lemma 3.5.1 and Lemma 3.5.2.

Theorem 3.5.1. *For iterative credit distribution in a strongly-connected weighted directed graph with the monotone-decreasing Δ'_k with k, if $\lambda^t \leq \Delta'_k/2$ in iteration t, the ranked list of users with top-k credits remain the same in subsequent iterations.*

Proof. The conclusion is composed of two parts. We begin with the first part, i.e., if $\lambda^t \leq \Delta'_k/2$ at the t-th iteration, then $x_1 > x_2 > \ldots x_k$. Consider the k- and $(k-1)$-ranked nodes. Since the relative WEC gap Δ'_k is monotone decreasing for k, we have

$$e'_k \leq \lambda^t \leq \Delta'_k/2 < \Delta'_{k-1}/2$$

Combined with $e'_{k-1} \leq \lambda^t$ in Lemma 3.5.1, we can get $e'_{k-1} < \Delta'_{k-1}/2$. In other words,

$$\begin{cases} e_k = |x_k - \pi_k| \leq \Delta_k/2, \\ e_{k-1} = |x_{k-1} - \pi_{k-1}| < \Delta_{k-1}/2. \end{cases}$$

By several operations, we have

$$x_{k-1} - x_k > ((\pi_{k-1} - \pi_k) - (\pi_k - \pi_{k+1}))/2 > 0$$

which holds since $\Delta_k/\pi_k < \Delta_{k-1}/\pi_{k-1} < \Delta_{k-1}/\pi_k$. Similarly, we can find that $x_{k-2} > x_k, \ldots, x_1 > x_k$. Moreover, if starting from $(k-1)$-ranked node (it holds as $e'_{k-1} \leq \lambda^t \leq \Delta'_{k-1}/2$), we have $x_{k-2} > x_{k-1}, \ldots, x_1 > x_{k-1}$ and thus $x_1 > x_2 > \ldots > x_k$.

Then we prove the second part, i.e., if $\lambda^t \leq \Delta'_k/2$ at the t-th iteration, then $x_k > x_j$, where j is from $k+1$ to n. Consider the k- and $(k+1)$-ranked nodes. We have

$$\begin{cases} e_{k+1} \leq \lambda^t \pi_{k+1} < \lambda^t \pi_k \leq \Delta_k/2, \\ e_k \leq \lambda^t \pi_k \leq \Delta_k/2. \end{cases}$$

Hence,

$$x_{k+1} - x_k < 0$$

and so for all other nodes with the rankings larger than k.

Since λ^t is geometrically decreasing for t, $\lambda^t < \Delta'_k/2$ holds for all the following iterations and so does the conclusion. □

Theorem 3.5.1 indicates that if there are no sybil users, Alg. 10 (or TrueTop) can generate the true top-K influential non-sybil users if $\lambda^t \leq \Delta'_K/2$, i.e., when $t \leq -\log(2K(\gamma-1))/\log(\lambda)$ or $t = O(|\log(K)/\log(\lambda)|)$ iterations. This also corresponds to the case of $\epsilon = 0$ with 100% accuracy. Since the total edge weights from/to the non-sybil region to/from the sybil region are relatively very small, we can expect that the sybil region has little impact on the influence rankings of non-sybil users. So the accuracy of TrueTop under sybil attacks is tightly related to how many sybil users can show up in the top-K list, i.e., the sybil-resilience of TrueTop, as analyzed in the following theorem.

Theorem 3.5.2. *Let α be the ratio of the total edge weight from the non-sybil region to the sybil region over the total edge weights in the non-sybil region. Assume that the attacker wants to place as many sybils into the top-K list as possible by retaining*

all the credits flowing into the sybil region. The number of sybil users in the top-K list after early termination in $t = O(\log(K)/\log(\lambda))$ iterations is upper-bounded by $K(1-(1-\alpha)^t)/(1-\alpha)^t$.

Proof. According to (60), the expected k-ranked WEC is

$$\langle \pi \rangle_k \approx \frac{\Gamma(k - \frac{1}{\gamma - 1})}{\Gamma(k)}$$

According to Proposition 3.4.1, the number of credits for $\mathcal{S}$ after the t-th iteration is given by:

$$C_{\mathcal{S}}^{(t)} = 1 - C_{\mathcal{H}}^{(t)} = (1 - \frac{1}{\alpha + \beta})\alpha(1 - \alpha - \beta)^{t-1} + \frac{\alpha}{\alpha + \beta}$$

The maximum $C_{\mathcal{S}}^{(t)}$ can be obtained as $1 - (1-\alpha)^t$ when $\beta \to 0$, i.e., the sybils conduct very limited interactions to the non-sybil users. Moreover, since the attacker wants to place as many sybils into the top-K list as possible, he can just divide the total credits $C_{\mathcal{S}}^{(t)}$ by the k-ranked WEC value. Then the number of sybils that own $\langle \pi \rangle_K$ credits is given by

$$n(K) = \frac{C_{\mathcal{S}}^{(t)}}{C_{\mathcal{H}}^{(t)} \langle \pi \rangle_K} \approx \frac{1 - (1-\alpha)^t}{(1-\alpha)^t \langle \pi \rangle_K}$$

Here we further approximate the $\langle \pi \rangle_K$. For the power law distribution, $2 \leq \gamma < 3$. Thus

$$\frac{\Gamma(k - \frac{1}{\gamma - 1})}{\Gamma(k)} > \frac{\Gamma(k-1)}{\Gamma(k)} = \frac{1}{k}$$

Hence, we can obtain $n(K) < K(1-(1-\alpha)^t)/(1-\alpha)^t$. □

Accordingly, we can easily derive the lower bound for the accuracy of TrueTop because there are at least $K(2 - 1/(1-\alpha)^t)$ true top-K non-sybil users in the final top-K list. Note that since $\alpha \ll 1$ and K is usually at the scale of 1,000 and 10,000, this upper bound is far less than K, meaning that there are only negligible sybil users in the top-K list.

3.6 Evaluation

In this section, we thoroughly evaluate the performance of TrueTop. We first introduce some implementation details and the runtime performance, followed by the datasets used in our evaluations. Next, we verify two underlying assumptions in our design. Finally, we evaluate the accuracy and sybil resilience of TrueTop under various sybil attacks.

3.6.1 Implementation and Runtime Performance

TrueTop is composed of two main components: the interaction graph construction and the credit distribution with early termination. We implemented both with a total of 2000+ lines of mixed code of Python and C++. Specifically, to efficiently handle the large-scale interaction networks (millions of nodes and billions of edges) in a commodity PC, we adopted the Graphchi computing framework (79) to implement the credit distribution of TrueTop. On our desktop with 3.4GHz Intel-i7 3770 CPU, 16G Memory, a 7200RPM hard disk, and Ubuntu 12.04 LTS, one single iteration of credit distribution took 0.3s, 2.5s, 9.2s, and 17.1s for our four datasets in Table 3.1 with 4K, 10K, 1M and 2M nodes, respectively. For a graph with 2M nodes, TrueTop can thus find the top-1000 influential users after 1,000 iterations within less than five hours on a commodity PC. Since TrueTop is expected to be run by a service provider with much more powerful computation resources, its runtime performance should be acceptable.

3.6.2 Datasets

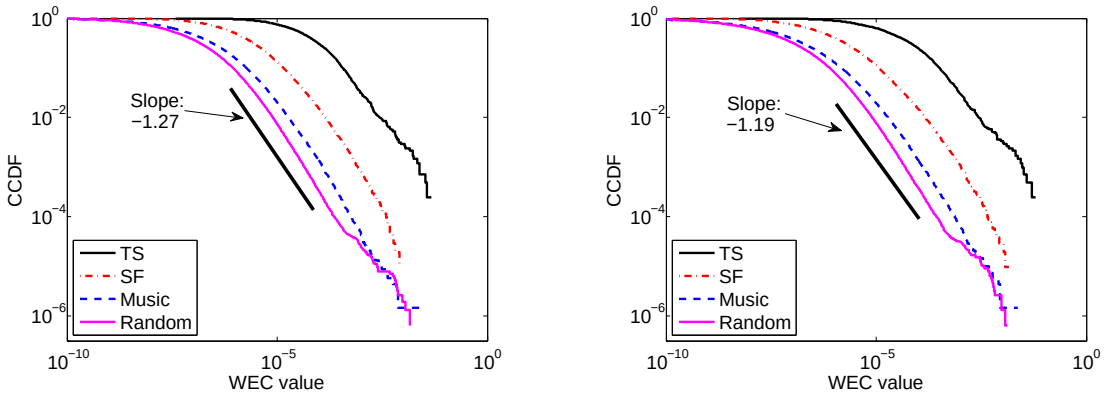
We crawled four representative datasets with public Twitter APIs. The `SF` and `TS` datasets include all the active users who have specified San Francisco Bay Area and

Table 3.1: Dataset Characteristics.

	SF	TS	Random	Music
Crawling period	8/30-11/30, 2013		6/28-9/28, 2013	
#users	176,506	5,827	1,999,834	999,807
#edges	1,493,924	40,031	63,803,204	34,688854
#users in GSCC	104,000 (58.9%)	4,127 (70.8%)	1,541,343 (77.1%)	687,693 (68.9%)
#edges in GSCC	1,305,834 (87.4%)	36,189 (90.4%)	55,781,520 (87.4%)	30,170,774 (87.0%)
#users in 2nd SCC	357	6	82	21

Tucson, Arizona in the location field of their public profiles in the crawling (or target) period, respectively. In addition, the `Random` dataset contains a random set of active Twitter users in the target period, and the `Music` dataset contains the active users who have used the keyword "music" in their tweets in the target period. Each dataset includes all the user IDs and also their time-indexed tweets during the target period, which include original tweets, retweets, replies, and mentions. Then we constructed two interaction graphs for each dataset according to the process in Section 3.4.2, one for sum-based edge weights and the other for entropy-based edge weights.

Table 3.1 summarizes the basic statistics of the interaction graphs of each dataset, which apply to both sum-based and entropy-based edge weights. As we can see, each interaction graph has a giant strongly connected component (GSCC) which is far larger than the second largest SCC. Since TrueTop measures user influence based on incoming interactions, the top-K influential users are in the GSCC with overwhelming probability. Our subsequent evaluations are thus done on the GSCC



(a) Sum-based (b) Entropy-based

Figure 3.2: The Distribution of WEC Values.

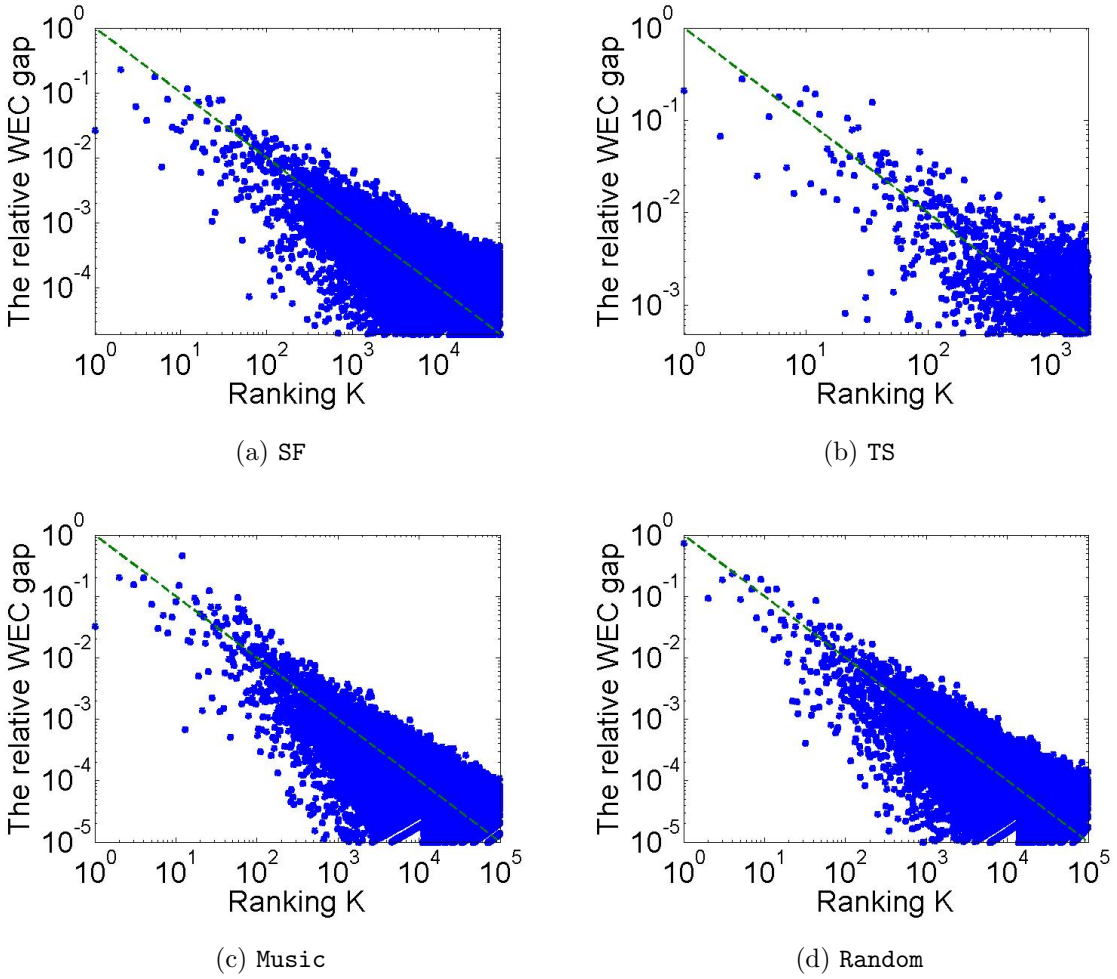
in each interaction graph only. We obtained very similar evaluation results for sum-based and entropy-based interaction graphs. Due to space limitations, we report the results for sum-based interaction graphs in most cases.

3.6.3 Feasibility Studies

WEC Value Characteristics

TrueTop bases its early termination of iterative credit distribution on two assumptions. First, the WEC values of non-sybil nodes follow a power-law distribution. Second, the relative WEC gap Δ'_k decreases as k increases. Now we verify these two assumptions.

Fig. 3.2 shows the log-log CCDF of the WEC values. We can see that all the CCDF curves are close to straight lines with the slopes from -2 to -1 for the WEC values larger than 10^{-6}. Since a power-law distribution with PDF $p(x) = (\gamma - 1)x^{-\gamma}$ has a CCDF $\bar{F}(x) = x^{1-\gamma}$, the WEC values of each interaction graph follow a power-law distribution with parameter γ from 2 to 3.



(a) SF (b) TS

(c) Music (d) Random

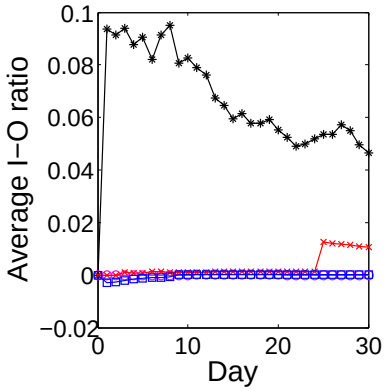
Figure 3.3: Relative WEC Gap Δ'_k.

Fig. 3.3 shows the log-log scale of Δ'_k as a function of k, where the results are shown up to $k = 10^5$ due to space constraints. We computed the WEC values by using $\nu = 10^{-4}$ as the error tolerance threshold of power iterations, which led to about 1,000 iterations. Δ'_k obviously decreases with an approximate slope of -1 in the log-log scale, which coincides well with the analysis in Lemma 3.5.2.

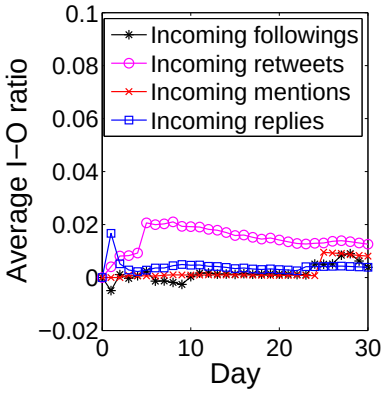
Interaction Analysis

Since there is no benchmark for the real-world sybils on Twitter, we designed an experiment to estimate the total edge weight from the non-sybil region to the sybil region in order to verify that it is relatively very small. To catch the growing intelligence of Twitter sybils, we adopted the behavior of the emerging social bots (101; 161; 52). Our experiment run as follows. We first purchased 1000 Twitter accounts, then divided them to mimic legitimate activities as in (101; 161), and finally investigated how many legitimate users will follow or interact with them. Specifically, we divided these 1000 accounts into five groups of equal size, each corresponding to a unique activity among following, tweeting, retweeting, mentioning, and replying. We ran the experiment for 30 days. In each day, we let each sybil user in each group initiate 10 activities corresponding to that group. For example, each sybil user in the Following group followed 10 randomly-chosen new users in each dataset every day. Except the sybil users in the Tweeting group, the sybil users in all the other groups initiated the corresponding activities only towards randomly chosen new users in each dataset. We also recorded the total followings/mentions/retweets/replies every sybil group received each day. In addition, we chose the `Random`, `SF`, and `Music` datasets as the target datasets in the first 14, middle 8, and last 8 days, respectively.

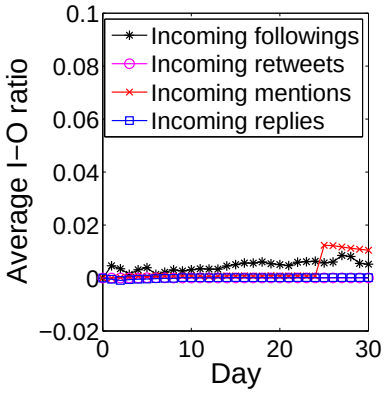
Fig. 3.4 shows the incoming-outgoing (I-O) ratios of each sybil group, which is defined as the number of total followings/mentions/retweets/replies each sybil group received every day over the total number of interactions initialized from the sybil group in the same day (i.e., 2,000). We have two observations. First, non-sybil users are very careful about whom to interact with and rarely interact with sybil users. Second, sybil users can get a non-trivial number of non-sybil followers. We manually found that most non-sybil followers are normal users out of reciprocity, social capi-



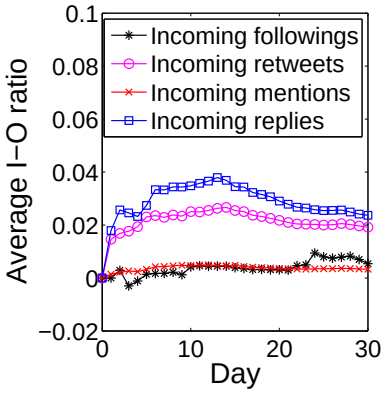
(a) Following



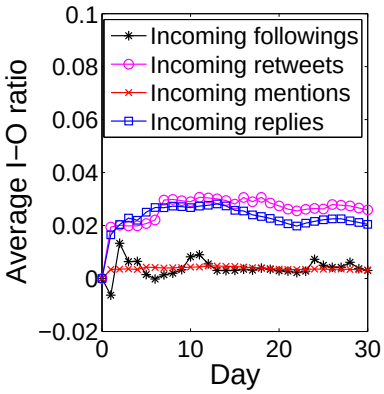
(b) Tweeting



(c) Retweeting



(d) Mentioning



(e) Replying

Figure 3.4: Incoming-outgoing Ratios for Sybil Groups, Where the Same Legend is Used in All the Figures.

Table 3.2: The Comparison of Incoming-outgoing Ratios Between Sybil and Non-sybil Communities Under Sum-based and Entropy-based Interaction Graphs.

Graph Model	Community	SF	Random	Music
Sum	Non-sybil	0.89	1.04	0.70
	Sybil	0.08	0.08	0.08
Entropy	Non-sybil	1.54	1.15	0.60
	Sybil	0.04	0.07	0.05

talists, or even spam accounts not suspended by Twitter, and this observation is in line with prior results in (59; 151). So incoming followings are less trustworthy for evaluating user influence than incoming replies, mentions, and retweets.

To compute the I-O ratios of the sybil and non-sybil communities, we randomly chose 30 groups of 200 users from each of `Random`, `SF`, and `Music` datasets. We then recorded the incoming and outgoing interactions of each non-sybil group every day in the same experimental period. The I-O ratio for each sybil or non-sybil group is redefined as the total incoming edge weight over the total outgoing edge weight. Table 3.2 compares the average I-O ratios of the sybil and non-sybil groups for both sum-based and entropy-based edge weights. As we can see, non-sybil communities always have much higher I-O ratios (i.e., much more balanced incoming and outgoing interactions) than sybil communities. Moreover, the entropy-based weight model yields lower and higher I-O ratios than the sum-based weight model for the sybil and non-sybil communities, respectively. We thus expect that the entropy-based weight model can lead to better sybil resilience than the sum-based model (as shown in Table 3.3).

3.6.4 Accuracy and Sybil Resilience Studies

Evaluation Methodologies

Since large-scale real experiments on Twitter inevitably violate the Twitter ToS, we resort to synthetic simulations to evaluate the accuracy and sybil resilience of TrueTop. We used all the four datasets and obtained quite consistent results. Below we show the evaluation results for the `SF` dataset only due to space constraints.

We modelled the strength of sybil attacks on Twitter by a parameter α, which refers to the ratio of the total edge weight in the non-sybil region over that from the non-sybil region to the sybil region. The default value of α, denoted by α^*, is obtained from our datasets as follows. Assume that the network is composed of a non-sybil region with n_1 twitterers and a sybil region with n_2 twitterers. According to our experiments, we found that about 0.98‰ of the users in the `SF` dataset have been suspended, so we set $n_1 = 1000n_2$. Moreover, assume that each non-sybil user initiate one interaction (i.e., retweeting, mentioning, or replying) to each of the other $n_1 - 1$ users, leading to $n_1(n_1 - 1)$ outgoing interactions. According to Table 3.2, the average I-O ratio of the non-sybil community for the sum-based interaction network is $(0.89 + 1.04 + 0.7)/3 \approx 0.88$. Therefore, the n_1 non-sybil users can receive about $1.88n_1(n_1 - 1) \approx 1.88n_1^2$ incoming and outgoing interactions. Similarly, the sybil users issue totally n_2n_1 interactions to the non-sybil region and receive about $0.08n_2n_1$ interactions from non-sybil users. We thus have the following approximation

$$\alpha^* = \frac{0.08n_2n_1}{1.88n_1^2} \approx 4.2 * 10^{-5}. \tag{3.5}$$

We used the following method to simulate the sybil region, which has been adopted in (155; 29). Given the interaction graph constructed from the `SF` dataset, we can expect that the majority of the 104,000 users there are non-sybil users, but we cannot tell which users are sybil or non-sybil users. So we manually attached to the original

interaction graph a sybil region which is a complete digraph of 500 sybil users and ran TrueTop over this augmented interaction graph. We assume the worst-case scenario in which the attacker aims to retain all the credits flowing into the sybil region, so there is no interaction from the sybil region to the non-sybil region. We then added w_g random links of weight one from the non-sybil region to the sybil region, which is equivalent to assuming that there are w_g accidental one-time interactions from non-sybil users to sybil users. w_g varied from 10 to 200 in our experiments. Since the total edge weight of the original interaction graph is about 10^6, we effectively simulated the parameter α from 10^{-5} to 2×10^{-4}. To simplify the presentation, we equate w_g with α and call it the attack strength as well hereafter.

We considered three strategies for the attacker to add the w_g links. In the *random attack*, the attacker randomly selects w_g users in the non-sybil region and adds a link of weight one from each to a randomly chosen user in the sybil region. In the *community attack*, the attacker performs a breadth-first search from a random user in the non-sybil region until w_g users are found, and it adds a link from each discovered user to a random user in the sybil region. In the *seed attack*, we fixed 10 seed users in the non-sybil region and assumed that the attacker knows all of them. The attacker performed a breadth-first search from the 10 seed users and randomly chose w_g users closest to any of the 10 seed users. It finally adds a link of weight one from each of them to a random user in the sybil region. Obviously, the seed attack corresponds to the strongest attack. We conducted 50 experiments for each attack and report the average result below. In addition, we chose 100 verified users as seed users in all simulations.

Now we introduce some metrics to measure the accuracy and sybil resilience. Recall that $\mathcal{U}_K$, $\mathcal{U}_K^*$, and $\tilde{\mathcal{U}}$ denote the TrueTop output, the true top-K influential users in the non-sybil region, and all the sybil users, respectively. We obtained $\mathcal{U}_K^*$

by running power iteration over the non-sybil region only with the error tolerance $\nu = 10^{-8}$. We measure the accuracy of TrueTop by comparing $\mathcal{U}_K$ and $\mathcal{U}_K^*$ via the following two types of errors.

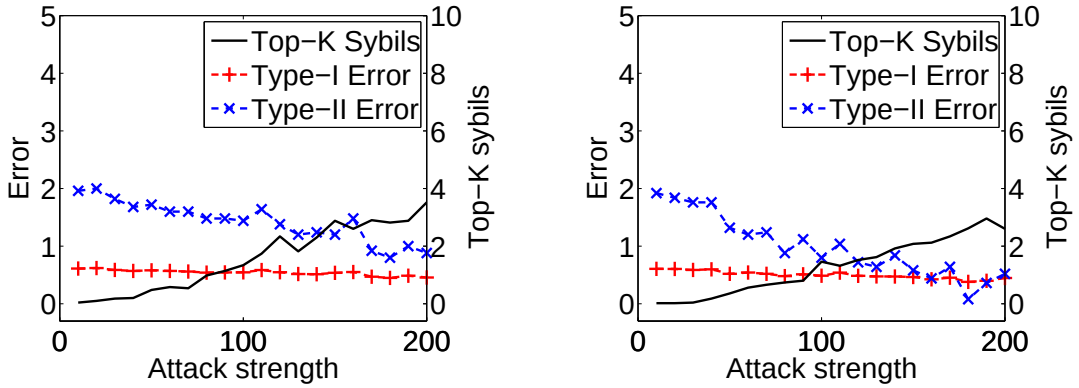
- Type-I error: $d(K)/K$, where $d(K)$ is the distance between $\mathcal{U}_K$ and $\mathcal{U}_K^*$ and computed according to Eq. (3.3). The metric measures the average rank offset of $\mathcal{U}_K^*$ from $\mathcal{U}_K$.
- Type-II error: $(K - |\mathcal{U}_K^* \cap \mathcal{U}_K|)$. This metric measures how many true top-K users are missed by TrueTop.

The sybil resilience of TrueTop is inversely proportional to $\#_{\mathsf{sybil}} = |\tilde{\mathcal{U}} \cap \mathcal{U}_K|$. After iterative credit distribution in TrueTop terminates, assumes that totally C credits are retained in the sybil region. Let $C_1, \ldots, C_K$ denote the credits of the top-K influential users in the non-sybil region in a non-decreasing order. Also assume that the attacker tries to maximize $\#_{\mathsf{sybil}}$ by arbitrarily manipulating the topology of the sybil region such that the C credits can flow into a few sybil users. We can derive $\#_{\mathsf{sybil}}$ as follows:

$$\#_{\mathsf{sybil}} = \begin{cases} 0 & \text{if } C < C_K, \\ \underset{1 \le x \le K}{\operatorname{argmax}} \quad C \ge x C_{K+1-x} & \text{else.} \end{cases}$$

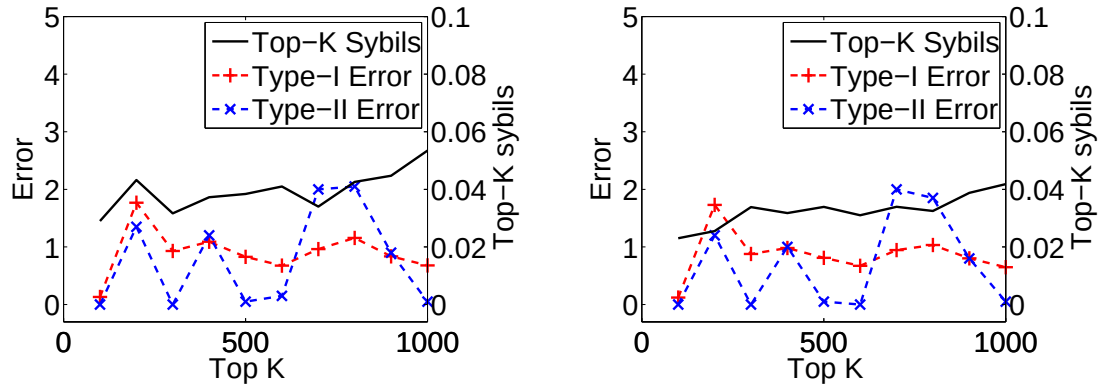
Basic Results

Fig. 3.5 shows the performance of TrueTop under different attack strengths in random and community attacks. In this experiment, we set $K = 100$ and $\epsilon = 0$. As the attack strength increases from 10 to 200, the type-I error is flat with less than one, and the type-II error is below two, both showing the high accuracy of TrueTop under different attack strengths. Moreover, the number of top-100 sybil users, i.e., $\#_{\mathsf{sybil}}$, slowly increases as w_g increases, which is as expected. $\#_{\mathsf{sybil}}$, however, stays below four for



(a) Random Attack (b) Community Attack

Figure 3.5: TrueTop Performance Under Different Attack Strengths

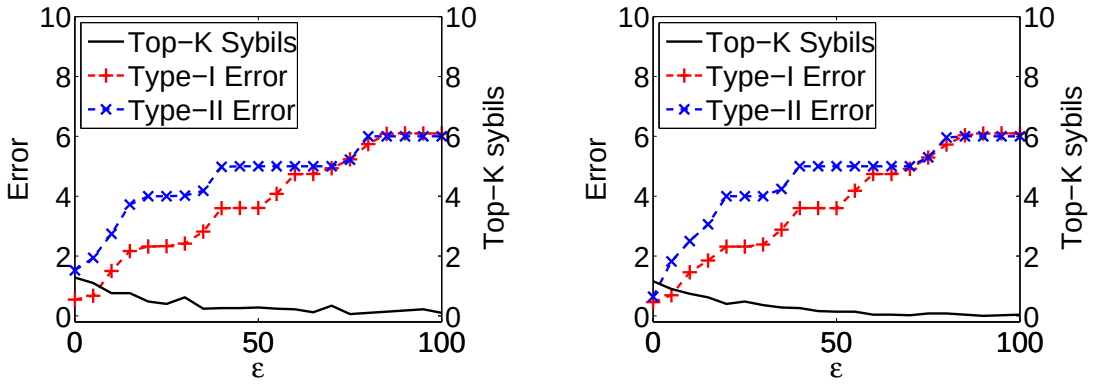


(a) Random Attack (b) Community Attack

Figure 3.6: TrueTop Performance for Different Ks.

both attacks. In addition, larger w_g is likely to increase the number of iterations and thus make the top-K list more accurate. So we can see that the type-II error overall decreases with increasing w_g.

Fig. 3.6 shows the performance of TrueTop under different Ks in random and community attacks. In this experiment, we set the $w_g = 100$ and $\epsilon = 0$. We also normalized $\#_{\text{sybil}}$ by K. Although $\#_{\text{sybil}}/K$ slowly increases with K due to more



(a) Random Attack

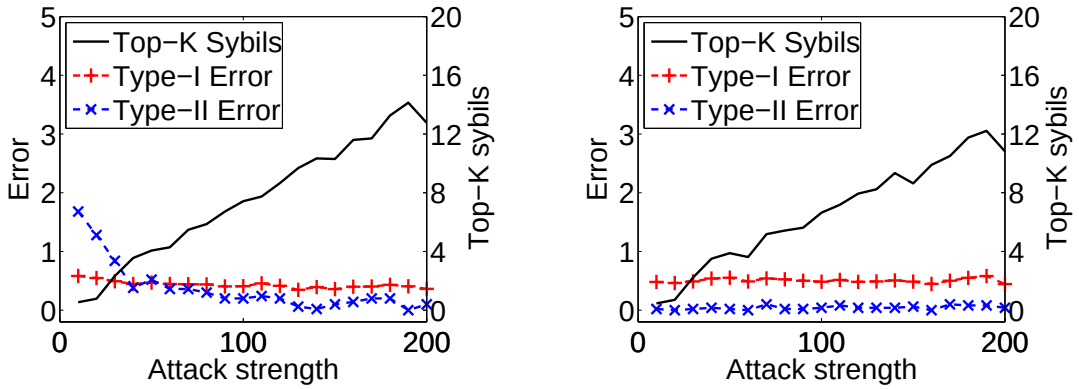
(b) Community Attack

Figure 3.7: TrueTop Performance Under Different ϵs.

iterations, it is always less than 6%. In addition, both type-I and type-II errors are always less than two, indicating the high accuracy of TrueTop.

Fig. 3.7 shows the performance of TrueTop under different ϵs in random and community attacks. In this experiment, we set $w_g = 100$ and $K = 100$. As expected, the larger the error tolerance ϵ, the larger both type-I and type-II errors. In contrast, $\#_{\text{sybil}}$ decreases with increasing ϵ due to fewer iterations towards credit distribution termination.

Fig. 3.8 shows the performance of TrueTop under seed attacks for both sum-based and entropy-based edge weights. In this experiment, we set $K = 100$ and $\epsilon = 0$. In addition, we randomly selected w_g users from $d = 3,000$ immediate successors of 10 random seed users, from which w_g links of weight one were added to the sybil region. We can have three observations from Fig. 3.8. First, TrueTop is still very accurate as both type-I and type-II errors are always less than 2. Second, seed attacks can yield more sybil users in the top-K list than both random and community attacks. Finally, entropy-based edge weights enable stronger sybil resilience than sum-based edge weights, as the former can dramatically increase the total edge weight in the



(a) Sum-based

(b) Entropy-based

Figure 3.8: Impact of Seed Attacks with Different Weight Models.

non-sybil region in contrast to the total edge weight from the non-sybil region to the sybil region. An effective defense against the seed attack is deferred to Fig. 3.11.

Table 3.3 shows the impact of design choices on the TrueTop performance. In this set of experiments, we set $K = 100$, $\epsilon = 0$, w_g from 10 to 200, and $d = 3,000$ for the seed attack. We compared the basic and reverse-WEC methods for seed selection, sum-based and entropy-based methods for determining edge weights, and also 10 versus 100 seed users. For simplicity, we added up the type-I errors, type-II errors, and $\#_{\text{sybil}}$ values under different attack strengths for each design choice, respectively. For each pair of design choices, we subtracted the sum of the second choice from that of the first one for the type-I error, type-II error, and $\#_{\text{sybil}}$, respectively. Since most results in Table 3.3 are positive, it is clear that the second choice in each pair can achieve higher accuracy and sybil resilience in most cases. Specifically, as expected, the entropy-based weight model yields better sybil resilience performance than the sum-based model.

Table 3.3: The Impact of Different Design Options on TrueTop Performance.

	Random attack			Community attack			Seed attack		
Seed selection: basic vs. rwec	0.11	0.232	-0.017	-0.03	-0.19	-0.192	0.11	0.19	0.316
Edge weights: sum vs. entropy	0.07	-0.002	0.226	0.08	0.00	0.572	-0.071	0.327	0.871
# of seeds: 10 vs. 100	4.26	0.099	0.122	2.89	0.121	0.078	3.66	0.136	2.8
	Type-I	Type-II	$\#_{\mathsf{sybil}}$	Type-I	Type-II	$\#_{\mathsf{sybil}}$	Type-I	Type-II	$\#_{\mathsf{sybil}}$

Comparison with Other Methods

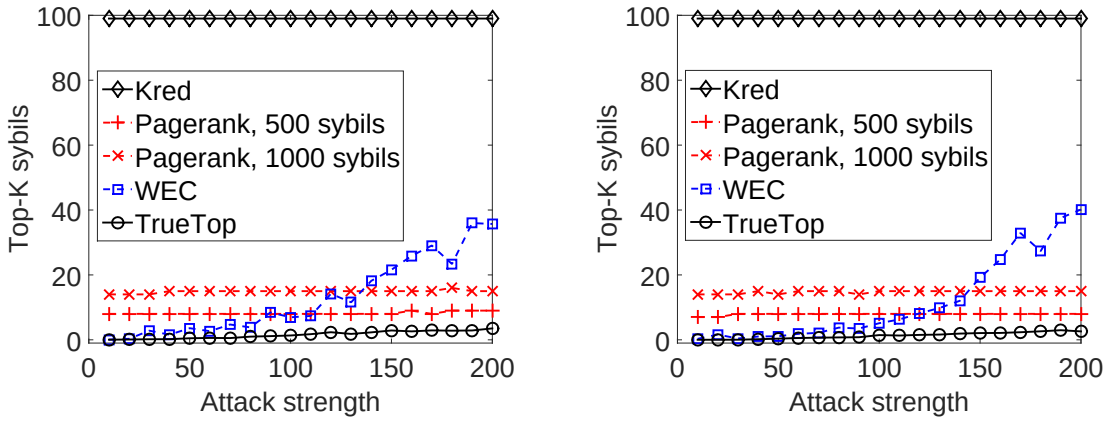
We compare our algorithm with the following methods.

1. *Kred* (2). Since Kred has publish its influence score algorithm on `http://kred.com/rules`, we select it as the benchmark mechanism. Kred only computes the influence score by how many interactions a user have received in the past 1,000 days. During our 90-days experiment, we let each of the 500 sybils retweet each other sybil once per day. Therefore, each sybil receives 44,910 interactions from the sybils in the end. We will see that this conservative attack is sufficient for filling the top-K list with mostly sybils.

2. *Pagerank* (113). One may think about using the Pagerank value of each user in the interaction graph to evaluate his influence. Modified power iteration with non-zero reset probability is commonly used to compute Pagerank values. We set the rest probability to 0.15.

3. *WEC by power iteration.* This method corresponds to TrueTop without early termination.

Fig. 3.9 compares the number of top-100 sybils of TrueTop with those of Kred, Pagerank and WEC by power iteration. As we can see, TrueTop allows less than 4 sybil users in the top-100 list under both random and community attacks. By comparison, the sybils in Kred can easily occupy 99 positions of the top-100 list. We also expect they will occupy all the top-100 positions if more interactions between the sybils were conducted. This is because the sybils can obtain unlimited incoming interactions from other sybils. Under WEC with power iteration, sybil users can occupy a significant portion in the top-100 list, as a lot more credits flow into and stay in the sybil region when power iteration terminates in contrast to TrueTop. In



(a) Under Random Attacks

(b) Under Community Attacks

Figure 3.9: Comparing TrueTop with Kred, Pagerank and WEC with Power Iteration Under the Random and Community Attacks.

addition, Pagerank leads to more top-100 sybil users than TrueTop and is less sensitive to the attack strength than WEC with power iteration. However, if we increase the number of sybil users from 500 to 1,000 without changing the attack strength, the top-100 sybil users under Pagerank will increase. This is because the more sybil users, the higher probability that credit distribution jumps to the sybil region due to resetting operations, the higher Pagerank values of some sybil users. So Pagerank is not sybil-resilient either, which is consistent with (32). In contrast, both TrueTop and WEC with power iteration are insensitive to the size of the sybil region.

Since WEC with power iteration is equivalent to seed-based iterative credit distribution without early termination, we also compare it with TrueTop with regard to the resilience to the seed attack. Note that Pagerank is not vulnerable to the seed attack because it does not use any seed user. Fig. 3.10 compares the top-100 sybil users of the two methods under the seed attack, where the number of immediate successors of the 10 victim seed users varies from $d = 5000$ to $10,000$ for the fixed attack strength

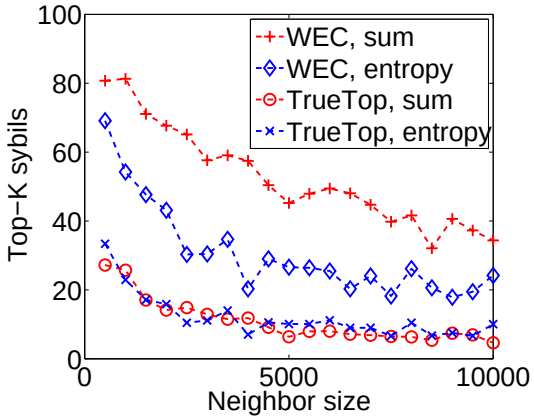


Figure 3.10: TrueTop and WEC under Seed Attacks.

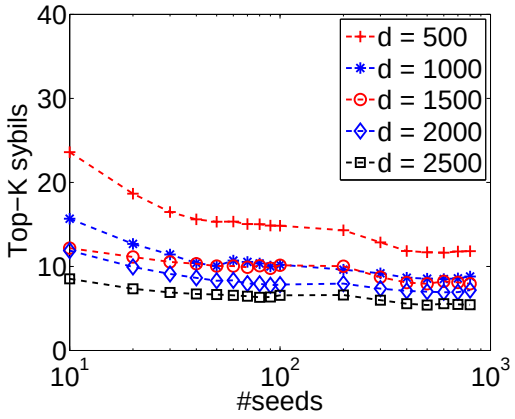


Figure 3.11: Defense Against the Seed Attack.

$w_g = 100$. As we can see, both methods yield more top-100 sybil users as d increases under sum-based and also entropy-based edge weights. This result is quite intuitive: the smaller d, the fewer nodes sharing the initial credits from the seed users, the more credits flowing into the sybil region over the w_g links, and vice versa.

An effective defense again the seed attack is to select more seed users and/or choose the verified users with more immediate successors as seed users. The efficacy of this defense is shown in Fig. 3.11. In this experiment, we assume that the attacker picked up 10 random seed users and then randomly selected d immediate successors of them for adding the w_g links to the sybil region. We varied the number of seeds from 10 to 800 for each value of d. As we can see, we can dramatically improve the resilience of TrueTop to the seed attack by increasing both the number of seed users and the number of immediate successors of the seed users.

Remarks

We have three remarks on the performance evaluation above. First, our evaluation results demonstrate the lower-bound performance of TrueTop. Specifically, we adopted

a very strong attacker model by assuming that the attacker withholds all the credits flowing into the sybil region by having zero interactions to the non-sybil region. In practice, sybil users often try to initiate interactions with non-sybil users for other purposes such as spamming and phishing than merely aiming to gain high influence scores. Therefore, we can expect fewer credits to stay in the sybil region than under our attacker model such that TrueTop shall have higher accuracy and sybil resilience in more practical settings. Second, we admit that our evaluations are not complete given so many design choices for TrueTop as shown in Table 3.3 and many possible attack strategies. We have only shown some important results here as the examples and expect similar results for other design choices and attack strategies. Finally, we modelled the sybil behavior in accordance with prior work (101; 161; 52). There are more advanced sybil attacks such as astroturfing (120) which could attract more legitimate interactions from non-sybil users. Unfortunately, there is no efficient way to simulate such advanced sybil attacks on a large scale. Instead, we use high attack strength w_g to model them in the experiment. As expected, TrueTop performs worse for higher w_g but still shows better performance in contrast to other methods. The performance of TrueTop will certainly degrade if the sybils could completely mimic the behavior of legitimate users, but manipulating the sybils to behave so intelligently will involve huge adversarial effort. TrueTop can thus significantly raise the bar for attacks on influence measurement.

3.7 Summary

Influential users are vital to accelerate large-scale information dissemination and acquisition on Twitter. In this chapter, we presented TrueTop, the first sybil-resilient system to measure the influence of Twitter users to the best of our knowledge. Our

theoretical studies and also performance evaluations confirmed the high accuracy and sybil resilience of TrueTop.

Chapter 4

YOUR ACTIONS TELL WHERE YOU ARE: UNCOVERING TWITTER USERS IN A METROPOLITAN AREA

4.1 Introduction

User privacy is arguably a major concern about Twitter. Specifically, user profiles and tweets may contain sensitive information about life, work, health, hobbies, political opinions, etc. Twitter currently offers little protection for user profiles and tweets which are virtually visible to anyone with or without an account. [1] Consequently, many users employ pseudonyms instead of real names in their profiles. In addition, Twitter users often hide their home locations (location for short thereafter), which are *permanent and static city-level* regions (e.g., Philadelphia) where most of their daily activities occur. Specifically, they may either not indicate their locations or report very general locations (e.g., state-level) in their profiles; they may not indicate their locations in their tweets either. For example, less than 34% of Twitter users explicitly specify their locations in their profiles (84), only 16% of Twitter users indicate city-level locations, and only 0.5% of tweets have a geo-tag (88).

There have been some efforts to infer a Twitter user's hidden location. Content-based methods (65; 33; 96; 88; 97) try to infer hidden locations based on geographic hints such as city landmarks in tweets. For example, a user who frequently mentions "Golden Bridge" in his tweets may indicate his location in the Bay Area. In contrast, network-based methods (15; 100; 73; 149; 38) leverage the fact that geographically-close people tend to form a connection or community in Online Social Networks

[1] Although Twitter allows a user to make his information visible to approved followers only, this privacy enhancement is rarely used in practice.

(OSNs) (117), so a user's location can be inferred from those of his online neighbors (or neighbors' neighbors, etc). Based on different estimation techniques, all these efforts (65; 33; 96; 88; 97; 15; 100; 73; 149; 38) seek to address the same question: how can we infer a Twitter user's hidden location from all his location-related tweets and/or OSN neighbors' locations?

This chapter targets a different and more challenging problem: is it feasible to efficiently discover the majority of Twitter users in any city-level metropolitan area ($\mathbb{A}$) without collaborating with Twitter? Since only 16% of Twitter users register city-level locations(88), it is infeasible to tackle our problem by directly checking users' tweets and profiles. In addition, directly applying any prior solution (65; 33; 96; 88; 97; 15; 100; 73; 149; 38) would inevitably involve checking every (255 million) Twitter user's tweets, followers, and/or followees, thus leading to a prohibitive cost.

An affirmative answer to our target problem above would have significant *positive* and *negative* impacts. On the *positive* side, finding the majority of the users in a specific area can not only benefit many applications such as local event detection and recommendation, business marketing, and emergency-alert dissemination, but also offer a feasible way to sample Twitter to facilitate the research concerning geographically related information. On the *negative* side, if an attacker can infer the majority of the Twitter users in a specific area, he could easily combine the location information with user tweets to better profile Twitter users who may or may not use pseudonyms, thus breaching their privacy and subjecting them to many identity-based attacks. Moreover, the Twitter users with exposed locations are vulnerable to large-scale location-based or geo-targeted spam campaigns (124).

In this chapter, we propose LocInfer, a novel and lightweight solution to the above problem for the first time in literature. The design of LocInfer is driven by two conjectures. First, a small but nontrivial fraction of users (15.9% on average in our datasets)

have specified a credible location in the target area $\mathbb{A}$ in their personal profiles, each of which is referred to as a *seed user* hereafter. Second, user communications in Twitter exhibit strong geographic locality in the sense that the users in the same area tend to interact more often than with those from outside. We confirm these two conjectures through large-scale datasets involving four representative metropolitan areas in U.S. Built upon these conjectures, LocInfer iteratively checks the immediate neighbors of the seed set, and the users who have tight connections with the seed set become new seeds and are added to the seed set. The final seed set contains the majority of Twitter users in $\mathbb{A}$ with overwhelming probability. LocInfer is highly efficient because only a small number of candidate users need to be checked in contrast to almost all the Twitter users when the existing methods (65; 33; 96; 88; 97; 15; 100; 73; 149; 38) are applied to our problem.

Our contributions can be summarized as follows.

- We motivate and formulate the problem of large-scale location inference, which is challenging given that only a small fraction of Twitter users have specified a credible city-level location in their personal profiles.
- We design LocInfer, a novel and lightweight solution that can uncover the majority of the Twitter users in a specific metropolitan area.
- We conduct extensive experiments to evaluate LocInfer using four large-scale datasets. Our results show that LocInfer can successfully discover on average 86.6% of the users with 73.2% accuracy.
- We propose a countermeasure against LocInfer for the Twitter users worrying about their location privacy and evaluate its effectiveness via simluations.

The rest of this chapter is organized as follows. Section 4.2 defines the problem.

Section 4.3 validates our two conjectures through four large-scale datasets. Section 4.4 details the LocInfer design. Section 4.5 evaluates LocInfer and our countermeasure. Section 4.6 surveys the related work. Section 4.7 summarizes the chapter.

4.2 Problem Statement, Terms and Notation

We use a directed and weighted multigraph [2] to model the diverse communications between Twitter users. In Twitter, people can follow others without mutual consent; they can mention others in their own tweets; they can also reply to or retweet others' tweets. We classify these communications into two categories: *following* and *interacting* (retweeting, replying, and mentioning), denoted by symbols $\mathcal{F}$ and $\mathcal{I}$, respectively. Such diverse communications are modeled as a directed and weighted multigraph $G = \langle V, E \rangle$, where each vertex $v \in V$ represents a user. We refer to a directed edge for the following type as a following edge and a directed edge for the interacting type as an interacting edge. A following edge $e_{ij}^{\mathcal{F}} \in E$ is formed when user i followed j; we call user i a *follower* of j and j a *followee* of i. In contrast, an interacting edge $e_{ij}^{\mathcal{I}} \in E$ is formed when user i mentioned, replied to, or retweeted j at least once; we call user i a *responder* of j and j an *initiator* of i. To model the interaction strength, we define $w(e_{ij}^{\mathcal{I}})$, the weight of edge $e_{ij}^{\mathcal{I}}$, as the total number of retweets, replies, and mentions from user i to j. For consistency, we also define the weight of any following edge as one. We use $N_I^{\mathcal{F}}(u), N_O^{\mathcal{F}}(u), N_I^{\mathcal{I}}(u), N_O^{\mathcal{I}}(u)$ to represent u's one-hop followers, followees, responders, and initiators, respectively. We also define the one-hop neighbors of u as $N(u) = N_I^{\mathcal{F}}(u) \cup N_O^{\mathcal{F}}(u) \cup N_I^{\mathcal{I}}(u) \cup N_O^{\mathcal{I}}(u)$.

Large-Scale Location Inference. Given a Twitter multigraph $G = \langle V, E \rangle$ and a target metropolitan area $\mathbb{A}$, we aim to obtain a target user list U which contains the majority of Twitter users in $\mathbb{A}$ without collaborating with Twitter.

[2] In a multigraph, two vertices may be connected by more than one edge.

Design goals. LocInfer is designed with the following goals.

- *High coverage.* The target user list U should cover the majority of Twitter users in $\mathbb{A}$. If we denote the actual Twitter users in $\mathbb{A}$ by U^*, the coverage can be computed as $|U \cap U^*|/|U^*|$.

- *High accuracy* [3] . The target users in U should be indeed located in $\mathbb{A}$. The accuracy can be computed as $|U \cap U^*|/|U|$.

- *Efficiency.* LocInfer should only involve checking Twitter users proportional in quantity to the population in $\mathbb{A}$ in contrast to existing methods (65; 33; 96; 88; 97; 15; 100; 73; 149; 38) which all need to check all the Twitter users. This efficiency requirement is particularly important because without Twitter's collaboration, the only free way to obtain the users' information is via third-party APIs, which is time-consuming as Twitter has strict rate limits on APIs invoking (13). For example, an authenticated user can only invoke the get-followers API 15 times per 15 minutes. Hence if we invoke this API once for each of the 255 million Twitter users, it will spend a single authenticated user about 485 years to obtain all the Twitter users' followers.

4.3 Conjectures Validation

As we mentioned in Section 4.1, LocInfer is built upon two important conjectures.

- *Conjecture 1*: A small but nontrivial fraction of users have specified a credible location in the target area $\mathbb{A}$ in their personal profiles.

[3]Note that *coverage* and *accuracy* correspond to the widely-used *recall* and *precision*, respectively. In this chapter we use the coverage and accuracy to make the meaning more straightforward in the context of user uncovering in an area.

Table 4.1: Seed Users in Four Metropolitan Areas in U.S.

Area $\mathbb{A}$	Population (rank in U.S.)	#Twitter users	#seed users (over #Twitter users)	#seeds with $\geq$ 1M followers
Tucson (TS)	996,544 (57th)	150,478	28,161 (18.65%)	0
Philadelphia (PI)	6,034,678 (7th)	911,236	144,033 (15.9%)	3
Chicago (CI)	9,522,434 (3rd)	1,437,888	318,632 (22.21%)	11
Los Angeles (LA)	16,400,000 (2nd)	2,476,400	300,148 (12.12%)	174

- *Conjecture 2*: User communications in Twitter exhibit strong geographic locality in the sense that the users in the same area tend to communicate more often than with those from outside.

In this section, we validate these two conjectures using four large-scale datasets.

4.3.1 Data Collection

We collect ground-truth Twitter users in different metropolitan areas by checking the self-reported locations in their profiles, a methodology that has been used to obtain the ground truth in (65; 33; 96; 88; 97; 15; 100; 73; 149; 38). Specifically, we use the Twitter geo-search API designed to return the recent or popular tweets in a specified *geo-circle* defined by latitude, longitude, and radius (13). For any interested area $\mathbb{A}$, we convert it into a geo-circle for the geo-search API, and we do not differentiate $\mathbb{A}$ and its corresponding geo-circle hereafter. The geo-search API returns the tweets from three types of users.

- *Geo-tagged users*: The users who recently published some tweets with a geo-tag in $\mathbb{A}$.

- *Geo-profiled users*: The users whose personal profiles containing a location in $\mathbb{A}$.

- *Retweeting users*: The users who recently retweeted some geo-tagged or geo-profiled users' tweets in $\mathbb{A}$.

Among them, we only use the geo-profiled users to build our datasets, because retweeting users are likely not in $\mathbb{A}$, and geo-tagged users may have just traveled to some places within the geo-circle instead of living there. Moreover, since the result of each geo-search API invoking corresponds to a random sampling of the active Twitter users, we keep invoking the geo-search API until no significantly more geo-profiled users can be discovered.

The self-reported locations have been found reliable (38), but the results from the geo-search API are still noisy for two reasons. First, the location descriptions in many users' profiles are ambiguous and arbitrary. For example, people living in Los Angeles may specify their locations as "South California", or "Los Angeles", or "LA", or just "CA." Second, the geo-search API often needs to covert a location description into a longitude-latitude pair for comparison with the specified geo-circle. Such conversions are often problematic and thus lead to wrong results. For example, when we searched the users in San Francisco Bay Area, the geo-search API returned some users in other places or even nonsense descriptions such as "somewhere you're not" and "wherever you not."

We thus refine the geo-profiled users as follows. For each user, we further verify whether his/her location description indeed contains a city name in $\mathbb{A}$. For this purpose, we first obtain the list of city names in $\mathbb{A}$ from the latest U.S. gazetteer data (5) and then compare the location description with the list. If there is an intersection, the user is considered a ground-truth user in $\mathbb{A}$.

4.3.2 Datasets

Using the above method, we collect user data in four metropolitan areas of Tuscon (Arizona), Philadelphia, Chicago, and Los Angeles. Our data collection ran from January to June 2014. Table 4.1 summarizes the four datasets. As we can see, the four populations vary from one million in `TS` to 16 millions in `LA`, from the not-so-popular areas (e.g., `TS`) to popular areas (e.g., `LA`). Note that all the metropolitan population information is from the U.S. Census Bureau.

4.3.3 Conjecture Validation

To validate the first conjecture above, we estimate the number of Twitter users for each area according to the eMarketer report claiming that 15.1% of U.S. people are using Twitter as of Feb. 2014 (50). As we can see from Table 4.1, the seed users range from 12.12% in `LA` to 22.21% in `CI` with the average ratio of 15.9%. This result is consistent with the measurement in (88) and implies that we have almost crawled all the users who have specified their city-level locations in these areas.

To validate the second conjecture above, we first define three locality metrics. In particular, for the multigraph $G = \langle V, E \rangle$ defined in Section 4.2, let V' denote any subset of V. We define follower locality $l_{\text{follower}}(V')$, followee locality $l_{\text{followee}}(V')$, and initiator locality $l_{\text{initiator}}(V')$ as

$$l_{\text{follower}}(V') = \frac{|N_I^{\mathcal{F}}(V') \cap V'|}{|N_I^{\mathcal{F}}(V')|}, l_{\text{followee}}(V') = \frac{|N_O^{\mathcal{F}}(V') \cap V'|}{|N_O^{\mathcal{F}}(V')|}, \\ \text{and } l_{\text{initiator}}(V') = \frac{w(N_O^{\mathcal{I}}(V') \cap V')}{w(N_O^{\mathcal{I}}(V'))}, \tag{4.1}$$

respectively, where $N_I^{\mathcal{F}}(V'), N_O^{\mathcal{F}}(V')$, and $N_O^{\mathcal{I}}(V')$ represent the followers, followees, and initiators of V', respectively, and $w(\cdot)$ represents the total weight of the corresponding interacting edges.

We let V' equal the seed users in each area and then compute the corresponding

Table 4.2: Locality in Each Area. Each Element is Composed of Three Values, Representing the Locality for the Seed Users in Each Area, the First Type of Random User Set, and The Second Type of Random User Set, Respectively.

$\mathbb{A}$	$l_{\text{follower}}(U)$ (%)	$l_{\text{followee}}(U)$ (%)	$l_{\text{initiator}}(U)$ (%)
TS	8.1 | 0.08|0.04	9.2 | 0.3 | 0.1	12.8 | 0.5 | 0.2
PI	4.9 | 0.4 | 0.2	8.4 | 1.5 | 0.6	14.9 | 2.7 | 1.2
CI	6.9 | 1.0 | 0.5	10.3| 3.3 | 1.3	16.9 | 5.2 | 2.6
LA	1.5 | 0.9 | 0.5	8.4 | 3.1 | 1.2	17.0 | 5.0 | 2.5

Table 4.3: Breaking Down the Initiator Locality by Three Types of Interactions.

$\mathbb{A}$	Replying (%)	Retweeting (%)	Mentioning (%)
TS	14.41	10.61	13.37
PI	14.05	12.95	16.99
CI	17.46	14.23	18.50
LA	15.43	15.44	19.05

locality. To do so, we crawl all the followers and followees of each seed user, and we also crawl the latest 600 tweets of each seed user to extract their initiators. For the comparison purpose, we build two types of random user sets. First, we merge the four seed sets into a single set from which we randomly select the same number of users as the seeds in each area. Second, we randomly select from the whole Twitter system the same number of users as the seeds in each area and compute their corresponding locality. We build 10 different user sets for both random user sets.

Table 4.2 shows the results of the locality analysis. We can see that the three locality values of the seed users in each area are always much higher than those of the

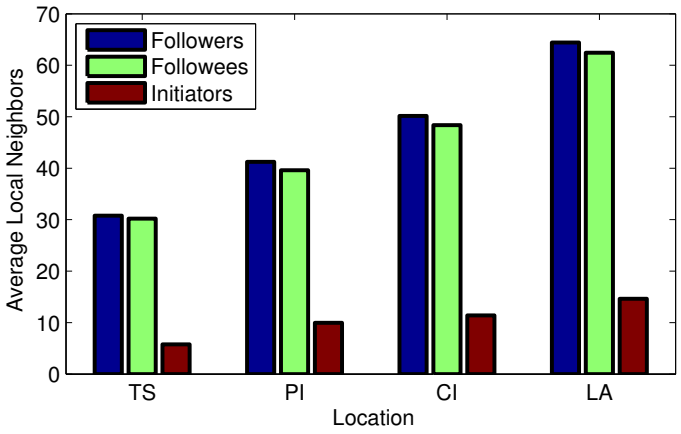


Figure 4.1: The Average Local Neighbors of the Seed Users.

random user sets. This result confirms our conjecture that physical proximity plays a big role in enabling online communications in Twitter. Moreover, Table 4.2 shows a higher percentage of a user's initiators in the same area than that of his/her followees. It is not surprising because a user may follow many people in different areas but often interact with only a few selected followees. In addition, we can see that the followee locality is much higher than the follower locality except in `TS`. The reason can be explained as follows. A celebrity user such as @rihanna can easily attract millions of followers from around the world, but she may only follow relatively fewer people. So we can expect a higher percentage of her followees in the same area (Los Angeles) than that of her followers. Since each of the areas except Tuscon has a large number of celebrity users, the followee locality is much higher than the follower locality. In contrast, Tuscon is a much smaller area with relatively few celebrity users, so we can expect similar followee and follower locality. Table 4.3 also shows that mentions, replies, and retweets contribute similarly to the initiator locality of each seed set, so we do not distinguish them in the LocInfer design.

Finally, although interacting communications (replies, mentions, and retweets) show much stronger locality than following communications, Fig. 4.1 shows that the

corresponding interacting edges (i.e., initiators) are much fewer than the following edges (i.e., followers and followees), meaning that people interact less than they follow others. Moreover, we also observe that people usually interact with the ones who they follow or follow them. In particular, let us define the overlap between $N_O^{\mathcal{I}}(V')$ and $N_I^{\mathcal{F}}(V') \cup N_O^{\mathcal{F}}(V')$ for each area as

$$\frac{|N_O^{\mathcal{I}}(V') \cap (N_I^{\mathcal{F}}(V') \cup N_O^{\mathcal{F}}(V'))|}{|N_O^{\mathcal{I}}(V')|}.$$

Our analysis shows that the average overlap for the four areas is 96.2%.

4.4 LocInfer

As stated before, our goal is to uncover the majority of Twitter users in an area $\mathbb{A}$. A naive solution is to use existing location inference methods (65; 33; 96; 88; 97; 15; 100; 73; 149; 38) for estimating the location of every Twitter user and then select the ones in $\mathbb{A}$. However, these methods are impratical for our problem. In particular, they would require crawling the followers, the followees, and many tweets for all the 255 million active Twitter users. Since Twitter has strict rate limits on data crawling (13), the crawling process for these methods will be time-consuming. In addition, the network-based methods (15; 100; 73; 149; 38) need to store and process the edges of the whole Twitter graph, thus leading to prohibitive storage and processing costs.

Now we present LocInfer, an efficient and effective three-step system to identify the majority of users in $\mathbb{A}$. As mentioned earlier, LocInfer is built upon two conjectures which have been experimentally validated in Section 4.3. First, we can find a nontrivial number (15.9% from our datasets) of users who have explicitly indicated a location in $\mathbb{A}$ through their personal profiles. These users are referred to as seed users (or seeds) in $\mathbb{A}$ and denoted by S. Second, user communications in Twitter exhibit strong geographic locality in the sense that users in the same area tend to

have more intensive communications with each other in Twitter than with those from outside. Based on these two conjectures, LocInfer first builds a seed set S (step 1 in Section 4.4.1) and then checks the one-hop neighbors of the seed set S, which constitute a candidate set denoted by C (step 2 in Section 4.4.2). Because of non-trivial seed set S and the strong geographic locality, C will cover the majority of the users in $\mathbb{A}$, but also include many users outside. Hence LocInfer chooses the candidate users who have tight connections with S as new seeds and add them to S, and this process continues until some termination conditions are met (step 3 in Section 4.4.3). The final seed set S contains the majority of Twitter users in $\mathbb{A}$ with overwhelming probability. LocInfer is highly efficient because it only checks a much smaller set of Twitter users in contrast to all the Twitter users if existing methods (65; 33; 96; 88; 97; 15; 100; 73; 149; 38) are applied.

We notice that many community structures (e.g., a group of people in different locations with common interests or past experience like classmates and colleagues) rather than the geographic community may also yield strong inter-connections. Hence LocInfer may include some users outside $\mathbb{A}$ in the candidate set C. However, the impact of such outside users is minimal because LocInfer only selects the users in the target area $\mathbb{A}$ as the seeds S and only chooses the target users who have strong communications with S later.

4.4.1 Step 1: Finding Seed Users

The first step in LocInfer is to extract the seed users who are most certainly in $\mathbb{A}$. To that end, we use the same method as in Section 4.3 by invoking the Twitter geo-search API to obtain the geo-profiled users and then refine them by checking their location descriptions to build the seed set S in $\mathbb{A}$.

It is possible that some people may specify the fake home locations in their profiles,

and it is infeasible to completely pinpoint and exclude such users. Fortunately, such self-reported locations have been verified to be very reliable (38) and have been used as the ground truth in (65; 33; 96; 88; 97; 15; 100; 73; 149; 38). Meanwhile, we may accidentally exclude some users indeed in $\mathbb{A}$, which is quite acceptable given our focus on obtaining a reliable seed set in this step. We admit that more advanced methods can be used for the seed searching and refinement, which are left for the future work.

4.4.2 Step 2: Finding Candidate Users

Based on the nontrivial number of seed users, the second step then is to construct a candidate-user set C from the one-hop neighbors of S that potentially covers the majority of Twitter users in $\mathbb{A}$ but is also much smaller than the set of all Twitter users. Below we first discuss how we decide the candidate users in C and then theoretically analyze the coverage of C.

Choosing C

We first build the candidate set C from the one-hop neighbors of S. The underlying intuition is based on the two conjectures validated in Section 4.3. Specifically, The second conjecture indicates that the users in the same geographic area tend to communicate more densely among themselves than to those from outside. On the one hand, if a user has very limited communications to all the seed users in S which occupies about 15.9% of the total users in $\mathbb{A}$, with high probability he/she is not in $\mathbb{A}$; on the other hand, a user that is indeed in $\mathbb{A}$ is very likely to have direct communication with some seeds. We therefore choose to build the candidate set C from the one-hop neighbors of S, denoted as $N(S)$.

Two details need further consideration. As defined in Section 4.2, each Twitter user has four kinds of neighbors in $G = \langle V, E \rangle$: followers, followees, initiators, and

responders. Which neighbors should we choose for each seed user? We observe from Fig. 4.1 that many Twitter users may follow a large number of other users, but they tend to subsequently interact with relatively few followees. Since people usually interact with the ones who they follow or follow them (with averagely 96.2% of overlap as stated in Section 4.3) and C should cover as many users as possible in $\mathbb{A}$, we consider all the followers and followees of each seed user in this step. Moreover, since each user in Twitter can follow arbitrary users without prior consent, the unidirectional following relationship is not a reliable indicator of geographic closeness. To deal with this issue, we propose to only select the candidate users to be the followers and followees of each seed user in S with each having at least t followees and t followers in S, where t is a system threshold.

More formally speaking, for each user $u \in N_O^{\mathcal{F}}(S) \cup N_I^{\mathcal{F}}(S)$, we compute $n_i^{\mathcal{F}}(u) = |N_I^{\mathcal{F}}(u) \cap S|$ and $n_o^{\mathcal{F}}(u) = |N_O^{\mathcal{F}}(u) \cap S|$. If both $n_i^{\mathcal{F}}(u)$ and $n_o^{\mathcal{F}}(u)$ are no less than t, user u is added to the candidate set C and ignored otherwise.

Alg. 2 implements the overall process. Specifically, we first create a followee counter and a follower counter for each user in $N_O^{\mathcal{F}}(S) \cup N_I^{\mathcal{F}}(S)$. Then we traverse the followee and follower list of each seed and increase the corresponding followee and follower counters. If both the followee and follower counters exceed t, we choose the user u as a candidate.

Coverage of C

The number of candidate users (i.e., $|C|$) is determined by both the number of seed users (i.e., $|S|$) and the system parameter t. A natural question is whether C can cover the majority of users in target area $\mathbb{A}$. It is important because the new seeds (or equivalently the target users) will be found only from C.

To analyze the coverage of C, we first define the following terms and notation.

Algorithm 2: Obtain the candidate set C by only checking the followee and follower lists of the seed set S.

input : $S, N_O^{\mathcal{F}}(S), N_I^{\mathcal{F}}(S), t$

output: the candidate set C

$C \leftarrow \varnothing$; $c_o[u] \leftarrow 0, \forall u \in N_O^{\mathcal{F}}(S)$; $c_i[v] \leftarrow 0, \forall v \in N_I^{\mathcal{F}}(S)$;
for $u \in S$ **do**
    $c_o[v] + +, \forall v \in N_O^{\mathcal{F}}(u)$; $c_i[v] + +, \forall v \in N_I^{\mathcal{F}}(u)$;
end
for $u \in N_O^{\mathcal{F}}(S)$ **do**
    if $c_o[u] \geq t$ *and* $u \in N_I^{\mathcal{F}}(S)$ *and* $c_i[u] \geq t$ **then**
        $C \leftarrow C + \{u\}$;
    end
end
return C.

We call users i and j *mutual followers* if they follow each other. Let $G_{\mathbb{A}} = \langle V_{\mathbb{A}}, E_{\mathbb{A}} \rangle$ be a subgraph of the Twitter multigraph $G = \langle V, E \rangle$, where $V_{\mathbb{A}} \subseteq V$ is the set of the Twitter users in the target area $\mathbb{A}$, and $E_{\mathbb{A}} \subseteq E$ is the set of the directed following edges among the users in $V_{\mathbb{A}}$. Consider a seed set $S \subseteq V_{\mathbb{A}}$ with $s = |S| = \alpha |V_{\mathbb{A}}|$ users, where $\alpha \in (0, 1]$. Let $N^t(S)$ denote the set of the followers and followees of S, each having at least t followers and t followees in S, where t is the system threshold stated before. The coverage ratio of C is defined as $r(t) = \frac{|N^t(S) \cup S|}{|V_{\mathbb{A}}|}$. We then have the following theoretical results about the coverage of C given $|S|$ and t.

Theorem 4.4.1. *Assume that each user in $V_{\mathbb{A}}$ has on average d_m mutual followers in $V_{\mathbb{A}}$. When $|V_{\mathbb{A}}|$ is large enough, the expected coverage ratio is $\overline{r}(t) \geq 1 - e^{-\alpha d_m}(1-\alpha) \sum_{i=0}^{t-1} \binom{s}{i} (\frac{p}{1-p})^i$, where $p = \frac{d_m}{|V_{\mathbb{A}}|-1}$.*

Proof. We first construct an undirected graph $G' = \langle V_{\mathbb{A}}, E' \rangle$, where an edge $e'_{ij} \in E'$ is formed if and only if users i and j are mutual followers. Let $N'^t(S)$ be the set of neighbors of S in G', each having at least t neighbors in S. We proceed to define the coverage of S in G' as $r'(t) = |N'^t(S) \cup S|/|V_{\mathbb{A}}|$.

We now compute $r'(t)$. Since each user has on average d_m edges in E', the probability of one user connecting to any other user is $p = \frac{d_m}{|V_{\mathbb{A}}|-1}$. Moreover, since there are $s = \alpha|V_{\mathbb{A}}|$ seed users, the probability of any non-seed node u connecting to less than t seed users in G' is given by

$$\rho = \sum_{i=0}^{t-1} \binom{s}{i} p^i (1-p)^{s-i} = (1-p)^s \sum_{i=0}^{t-1} \binom{s}{i} (\frac{p}{1-p})^i \ . \tag{4.2}$$

When the number of users in $V_{\mathbb{A}}$ is large, we have

$$\begin{aligned} \lim_{|V_{\mathbb{A}}|\to+\infty}(1-p)^s &= \lim_{|V_{\mathbb{A}}|\to+\infty}(1-d_m/|V_{\mathbb{A}}|)^{\alpha|V_{\mathbb{A}}|} \\ &= e^{-\alpha d_m} \ . \end{aligned}$$

Since there are $|V_{\mathbb{A}}|-s$ non-seed users, the expected number of non-seed users connecting to t or more seeds in S can be computed as $(|V_{\mathbb{A}}|-s)(1-\rho) = |V_{\mathbb{A}}|(1-\alpha)(1-\rho)$. When $|V_{\mathbb{A}}|$ is large, we have

$$\begin{aligned} \overline{r'}(t) &= |N'^t(S) \cup S|/|V_{\mathbb{A}}| \\ &= 1-(1-\alpha)\rho \\ &\approx 1-(1-\alpha)e^{-\alpha d_m} \sum_{i=0}^{t-1} \binom{s}{i} (\frac{p}{1-p})^i. \end{aligned}$$

Since each edge in E' corresponds to two directed edges in E, all the users in $N'^t(S)$ must belong to $N^t(S)$. On the other hand, a user in $N^t(S)$ may not appear in $N'^t(S)$. For example, consider a user who has exactly t followers and t followees in S in graph G, where none of his followers and followees are the same. Then this user is an isolated vertex in G', and he is certainly in $N^t(S)$ but not in $N'^t(S)$. Therefore, we have $N'^t(S) \subseteq N^t(S)$ and $\overline{r'}(t) \leq \overline{r}(t)$, and the theorem is proved. □

Corollary 4.4.1. $\overline{r}(t=1) \geq 1 - e^{-\alpha d_m}(1-\alpha)$.

Corollary 4.4.2. $\overline{r}(t=2) \geq 1 - e^{-\alpha d_m}(1-\alpha)(1+\alpha d_m)$.

Since $|V_{\mathbb{A}}|$ is often large in practice, Theorem 4.4.1 indicates that the coverage ratio $\overline{r}(t)$ approaches 1 when αd_m is large enough. Moreover, the choice of t involves a tradeoff between the crawling cost and the coverage. Specifically, the larger the t, the fewer the candidates in C, the smaller the crawling cost, the more likely to miss some users in $\mathbb{A}$ (i.e., the lower coverage), and vice versa. The size of S also affects the choice of t. On the one hand, if S constitutes a relatively large portion of the users in $\mathbb{A}$ (say, $\alpha = 30\%$), it may be safe to use larger t because many users in $\mathbb{A}$ are more likely to have more followees and followers in S. On the other hand, if S constitutes a relatively small portion of the users in $\mathbb{A}$ (say, $\alpha = 10\%$), it may be safe to use smaller t to avoid excluding too many users in $\mathbb{A}$.

Here we illustrate how many seeds are needed to achieve a nearly 100% coverage. Assume that each user in $\mathbb{A}$ has on average 15 mutual followers (i.e., $d_m = 15$). According to Corollaries 4.4.1 and 4.4.2, when $t = 1$, 20% of the users as seeds can cover 96.02% of the target users in $\mathbb{A}$, and when $t = 2$, 20% and 30% of the users as seeds can cover 84.07% and 95.72% of the users in $\mathbb{A}$, respectively. Similarly, if $d_m = 30$, only 10% and 15% of the users as seeds can cover 95.52% and 95.99% of the users for $t = 1$ and $t = 2$, respectively. These results indicate that when each user has sufficient mutual followers in $\mathbb{A}$, the followers and followees of a small number of seeds can cover the majority of the target users in $\mathbb{A}$.

4.4.3 Step 3: Finding Target Users U

Although the candidate set C covers nearly all the users in $\mathbb{A}$ for proper t, it may contain many users not in $\mathbb{A}$ who nevertheless have at least t followees and also t

followers in the seed set S. For example, social butterflies (151) or social capitalists (59) have been reported to automatically follow back whoever follows them, and users may also follow each other due to reciprocity (151; 59). We thus design the next step to identify the target user set U in $\mathbb{A}$ from C using both the following and interacting connections among the users.

Our key observation as stated is that each target user is very likely to demonstrate significant locality with the seed user set S. In other words, we expect that the target users form a strong local community with the seed users. From the initial seed set S, we iteratively check the candidate users in C, and the candidate who has the highest locality value with the seeds becomes a new seed and is added to S. The process iterates until certain conditions are met.

How should we compute the locality of Twitter users with diverse communications? Inspired by the Eq. (4.1), we consider three types of locality for any candidate user $u \in C$: follower locality $l_{\text{follower}}(u)$, followee locality $l_{\text{followee}}(u)$, and initiator locality $l_{\text{initiator}}(u)$, which are computed as

$$\begin{aligned} l_{\text{follower}}(u) = \frac{|N_I^{\mathcal{F}}(u) \cap S|}{|N_I^{\mathcal{F}}(u)|}, l_{\text{followee}}(u) = \frac{|N_O^{\mathcal{F}}(u) \cap S|}{|N_O^{\mathcal{F}}(u)|}, \\ \text{and } l_{\text{initiator}}(u) = \frac{w(N_O^{\mathcal{I}}(u) \cap S)}{w(N_O^{\mathcal{I}}(u))}, \end{aligned} \tag{4.3}$$

where $N_I^{\mathcal{F}}(u), N_O^{\mathcal{F}}(u)$, and $N_O^{\mathcal{I}}(u)$ are u's followers, followees, and initiators, respectively, and $w(\cdot)$ denotes the total weight of the corresponding interacting edges.

We also consider two methods to integrate the three types of locality. First, we choose the maximum one among them as u's locality, i.e.,

$$l(u) = \max\{l_{\text{follower}}(u), l_{\text{followee}}(u), l_{\text{initiator}}(u)\} \ . \tag{4.4}$$

Second, their weighted combination is used as the locality of u, i.e.,

$$l(u) = \epsilon_1 l_{\text{follower}}(u) + \epsilon_2 l_{\text{followee}}(u) + \epsilon_3 l_{\text{initiator}}(u) \ , \tag{4.5}$$

where $0 \leq \epsilon_1, \epsilon_2, \epsilon_3 \leq 1$ and $\epsilon_1 + \epsilon_2 + \epsilon_3 = 1$. In this report, we choose each of them to be $1/3$ for simplicity and leave other possible assignments as the future work.

Finally, we iteratively find the target users based on one of their five types of locality with regard to the seed set S. In each iteration, we compute the locality for each candidate $u \in C$ according to Eq. (4.3), Eq. (4.4), or Eq. (4.5). The candidate with the highest locality is removed from C and added to S as a new seed, as this user contributes most to the tightness of the community around S. In addition, the follower, followee, and/or initiator locality values of the remaining candidates in C need be updated in every iteration. Here we just use the followee locality to illustrate the updating operation. Let $l^{(m)}_{\text{followee}}(u)$ denote the followee locality for candidate u in iteration $m \geq 0$, where $l^{(0)}(u)$ can be computed by using the initial seeds in S. Assuming that u^* has been chosen as a new seed in iteration m, we update the followee locality for candidate u as

$$l^{(m+1)}(u) = \begin{cases} l^{(m)}(u) + 1/|N_O^{\mathcal{F}}(u)| & \text{if } u^* \in N_O^{\mathcal{F}}(u), \\ l^{(m)}(u) & \text{o.w.} \end{cases} \tag{4.6}$$

Follower and initiator locality can be updated similarly, and we may need to update the overall locality according to Eq. (4.4) or Eq. (4.5). The iteration terminates when the seed set S contains a desired number of users in $\mathbb{A}$, denoted by τ_A. Then the sought target users correspond to all the users in $\mathbb{A}$. The complete process is summarized in Alg. 3, which is implemented using a max-priority queue (39).

The termination threshold $\tau_{\mathbb{A}}$ can be chosen in two ways. First, we can set $\tau_{\mathbb{A}}$ as the estimated number of Twitter users in $\mathbb{A}$, e.g., about 15.1% of the population in $\mathbb{A}$ if $\mathbb{A}$ is in U.S. (50). Second, τ_A can be chosen according to the level of confidence we desire. In particular, our algorithm essentially ranks all the candidate users according to our confidence about their locations in $\mathbb{A}$. The later a candidate user is added to U, the lower confidence we have that he is indeed in $\mathbb{A}$. Therefore, if we want to

Algorithm 3: Identify target users in $\mathbb{A}$ from C.

input : $S, C, \tau_{\mathbb{A}}$

output: U, i.e., the users in $\mathbb{A}$

1 $U \leftarrow S$;

2 Compute $l(u), \forall u \in C$, according to Eq. (4.3), (4.4) or (4.5);

3 $Q \leftarrow \emptyset$;

4 **for** $u \in C$ **do**

5 $\quad$ INSERT(Q, u);

6 **while** $|U| < \tau_{\mathbb{A}}$ **do**

7 $\quad$ $u^* \leftarrow$ EXTRAC–MAX(Q);

8 $\quad$ $U \leftarrow U + \{u^*\}, S \leftarrow S + \{u^*\}$;

9 $\quad$ **for** $u \in N_I^{\mathcal{F}}(u^*)$ **do**

10 $\quad\quad$ INCREASE–KEY$(Q, u, l(u) + 1/|N_O^{\mathcal{F}}(u)|)$;

11 **return** U.

obtain a set of target users in $\mathbb{A}$ with high confidence, a small $\tau_{\mathbb{A}}$ should be used; if we want to cover more users in $\mathbb{A}$, a larger $\tau_{\mathbb{A}}$ is suitable.

We now analyze the complexity of Alg. 3. In Lines 4-5, we build a max-priority queue Q based on each candidate's locality value, of which the complexity is $\mathcal{O}(|C| \log |C|)$. The loop beginning from Line 6 is used to find the target user one at a time. In each iteration, we extract the maximum value from the priority queue Q in Line 7, set it as a new seed in Line 8, and update the locality value of all its followers in Lines 9-10. The complexity of Line 6-10 is $\mathcal{O}(\tau_{\mathbb{A}} d \log(|C|))$, where d is the average degree in $\mathbb{A}$. Hence the overall complexity of Alg. 3 is $\mathcal{O}((|C| + \tau_{\mathbb{A}} d) \log(|C|))$.

One may wonder why we do not add more candidates to C once a candidate is added as a new seed to S. We have shown in Section 4.4.2 that the candidate

users discovered through the initial seed set S cover the majority of users in $\mathbb{A}$ with overwhelming probability. It is thus unlikely that we can identify more candidate users from newly identified seeds, which has been validated by our simulations in Section 4.5.3. We thus choose not to add more candidates in each iteration.

4.4.4 Cost Analysis

We now analyze the cost of LocInfer, which consists of the crawling cost and computation cost, and briefly compare it with the existing methods.

We first analyze the crawling cost of LocInfer, which is important given the tight rate limitations Twitter enforces on data crawling. First, Step 1 in LocInfer involves invoking the Twitter geo-search API continuously to obtain the initial seed set S and needs to crawl some geo-tagged users' tweets. Second, Step 2 requires crawling the followees and followers of each seed user in S. Finally, Step 3 needs to crawl the followees, followers, and initiators of each candidate user in C. Recall that d denotes the average number of followers and followees each seed user has. Our datasets in Section 4.3 show that d is approximately 600. It has also been reported that 15.1% U.S. people use Twitter (50) and that 15.9% of Twitter users report city-level locations and become seeds in LocInfer. In LocInfer, a user is chosen as a candidate if he has t followers and t followees in S. So we can expect that the candidate set size $|C|$ is much smaller than $d|S|$, i.e., 14.4 times the population in the target area $\mathbb{A}$. In contrast, all previous (potential) solutions (65; 33; 96; 88; 97; 15; 100; 73; 149; 38) involve crawling all the Twitter users. Thus LocInfer has a much smaller crawling cost, which makes it practical.

The computation cost of LocInfer is dominated by the third step with the complexity of Alg. 3 being $\mathcal{O}((|C| + \tau_{\mathbb{A}} d) \log(|C|))$, where d is the average neighbors of each user and $\tau_{\mathbb{A}}$ is the number of target users in $\mathbb{A}$.

4.4.5 *Countermeasure*

LocInfer aims to discover the majority of users in any target area even if many of them do not disclose their locations explicitly in their personal profiles. We propose a simple countermeasure here to alleviate the possible concerns of some sensitive users about their location privacy. Since LocInfer discovers a user's location based on his tight connections with other users in the same area, the user can effectively hide his home location by following, retweeting, mentioning, and replying Twitter users outside his home area on a regular basis. This strategy is meaningful because people can follow or interact with others who are in different areas but share the same interests. For example, a user in New York City and the other in Los Angeles may interact in Twitter because they were university classmates in Dallas or knew each other in a concert. The efficacy of this countermeasure is evaluated in Section 4.5.5.

4.5 Performance Evaluation

In this section, we thoroughly evaluate LocInfer. As stated before, this chapter targets a different problem with existing work (65; 33; 96; 88; 97; 15; 100; 73; 149; 38), and hence we will not compare LocInfer with them head to head but could incorporate with them in our future work.

4.5.1 *Methodology*

To evaluate LocInfer, we first need build a testing multigraph $G = \langle V, E \rangle$ formed by both users known to be and not be in a target area $\mathbb{A}$, where one challenge is that we cannot directly determine all the Twitter users in $\mathbb{A}$.

To tackle this challenge, we adopt the method used by existing work (65; 33; 96; 88; 97; 15; 100; 73; 149; 38). Specifically, since the self-reported locations have been

found reliable (38), for each area $\mathbb{A}$ in Table 4.1, we treat all the seed users in S discovered in the first step as the positive ground truth (i.e., they are indeed in $\mathbb{A}$) and randomly partition S into a seed subset $\overline{S}$ of size $\alpha|S|$ and a testing subset T of size $(1-\alpha)|S|$.

For the negative ground truth, we check the followers and followees of S and record the set of users who have specified a location outside $\mathbb{A}$ and randomly choose β fraction of these users, where β is set as the ratio of seed users over the estimated number of Twitter users in $\mathbb{A}$, as shown in the fourth column of Table 4.1. We denote by Θ the resulting user set and let $V = S \cup \Theta$. We finally compute edges among all the users in V according to their followings and interactions by analyzing their followers, followees, and the latest 600 tweets.

We then apply LocInfer to the testing multigraph G. Specifically, we first use $\overline{S}$ as the seed set and apply Alg. 2 to generate the candidate set C. We then apply Alg. 3 to C to generate the target user set U by choosing a $\tau_{\mathbb{A}}$. Following the definitions in Section 4.2, the *coverage* can be computed as $|U \cap S|/|S|$, and the *accuracy* can be computed as $|U \cap S|/|U|$ ($|U| = \tau_{\mathbb{A}}$).

Unless stated otherwise, we choose $t = 2$ when building the candidate set C with Alg. 2 for `LA` and $t = 1$ for all other three datasets, and set $\alpha = 0.159$, the average ratio for the four datasets in Table 4.1. The testing multigraphs are summarized in Table 4.4.

4.5.2 Accuracy

We first evaluate the accuracy of LocInfer. We compute five locality values for each user, including follower locality, followee locality, initiator locality, and the two locality values defined in Eq. (4.4) and Eq. (4.5), respectively.

Fig. 4.2 shows the accuracy of LocInfer for the four datasets, where $\alpha = |\overline{S}|/|S| =$

Table 4.4: The Testing Multigraphs for the Evaluation. ($\alpha = 0.159$)

$\mathbb{A}$	$\lvert S\rvert$	$\lvert\overline{S}\rvert$	$\lvert T\rvert$	β	$\lvert\Theta\rvert$
TS	28,161	4,478	23,683	18.65%	162,446
PI	144,033	22,901	121,132	15.9%	630,321
CI	318,632	50,662	267,970	22.21%	1,529,431
LA	300,148	47,724	252,424	12.12%	710,085

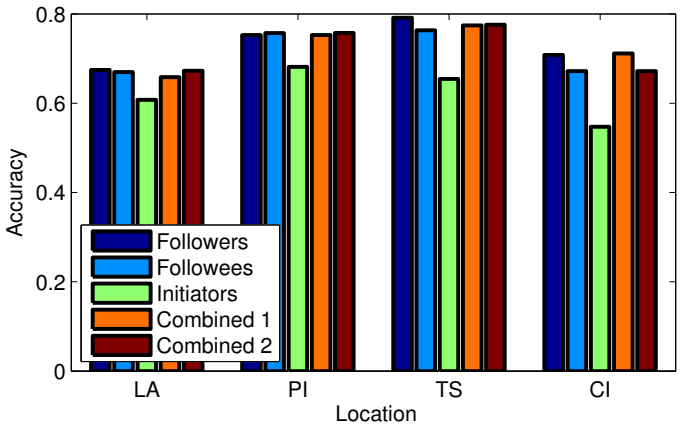
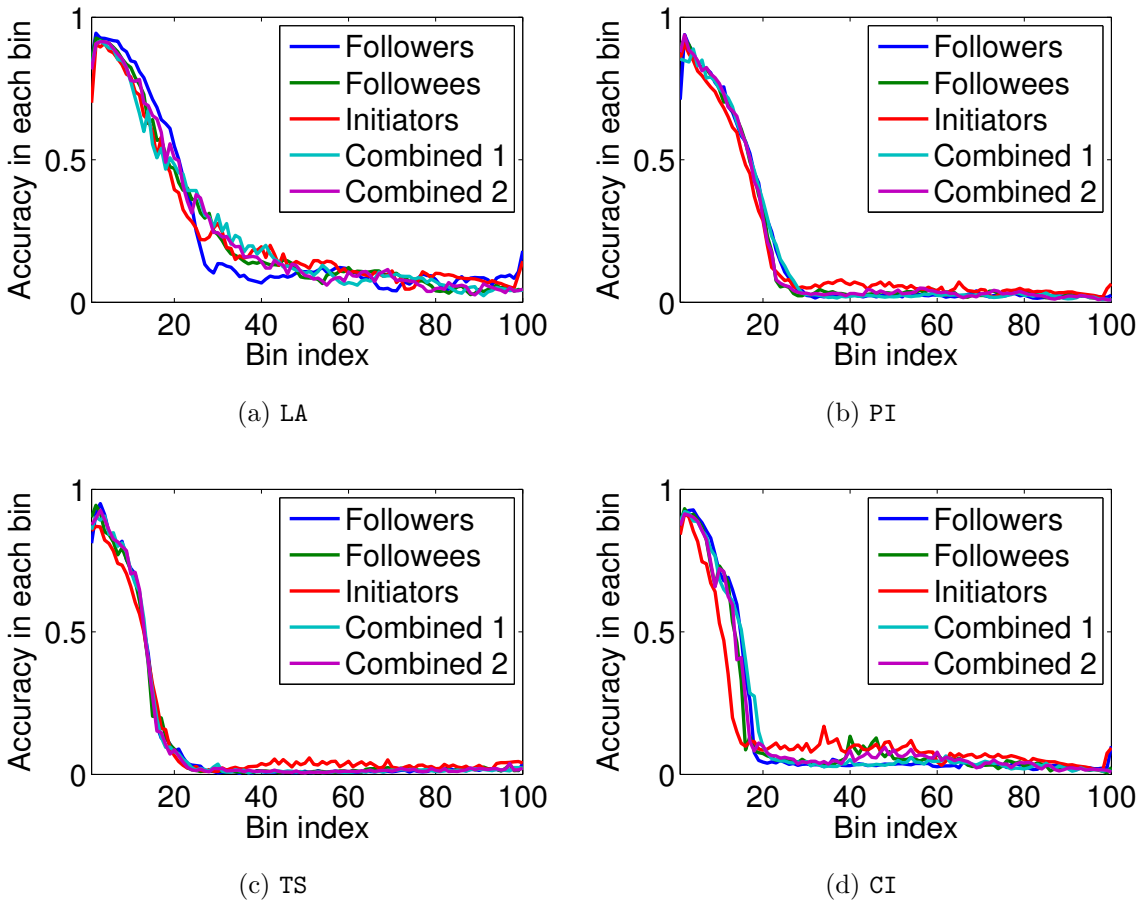


Figure 4.2: The Accuracy of LocInfer.

0.159 and $\tau_{\mathbb{A}} = |S|$. We can see that the five locality metrics all lead to high accuracy in each area, and initiator locality has the worst performance among them. Specifically, the average accuracy of four datasets for each locality are 73.2%, 72.6%, 62.3%, 72.4%, 71.9%, respectively. The reason is that initiator locality depends on interacting edges (corresponding to replies, mentions, and retweets) which are much sparser than following edges in the directed Twitter multigraph as shown in Fig. 4.1. Therefore, if many users in $\mathbb{A}$ only follow many people but do not interact with them subsequently, they may be reachable from seed users through following edges but not from interacting edges. We will show the coverage for different locality metrics in the following Section 4.5.3. Moreover, the locality defined in Eq. (4.4) and Eq. (4.5)



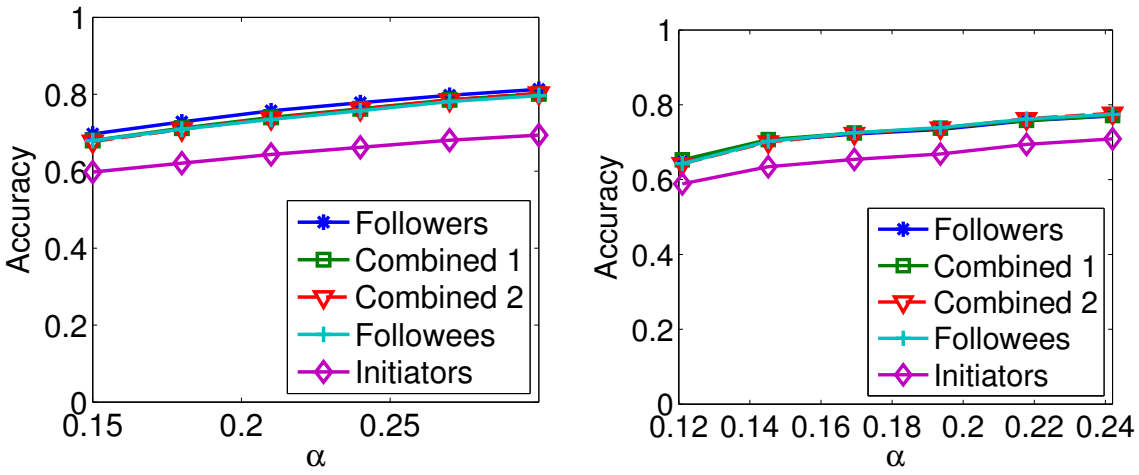
(a) LA (b) PI

(c) TS (d) CI

Figure 4.3: Detailed Accuracy Illustration.

have nearly the same accuracy with both the follower and followee locality. This is expected because about 96.2% of the seed set's initiator neighbors are from their followers or followees, as indicated in Section 4.3.

To shed more light on the accuracy of LocInfer, we set $\tau_A = |C|$ so that $U = C \cup \overline{S}$ when Alg. 2 terminates, i.e., every candidate user is eventually added into S. Let U' denote the newly discovered users (may not in $\mathbb{A}$), i.e., $U' = C$. We partition U' into 100 bins of equal size $|U'|/100$ according to the order they are added, where the bins of smaller indexes contain the users discovered earlier. Let x_i denote the number of positive ground-truth users in the i-th bin. Fig. 4.3 shows the accuracy of the i-th



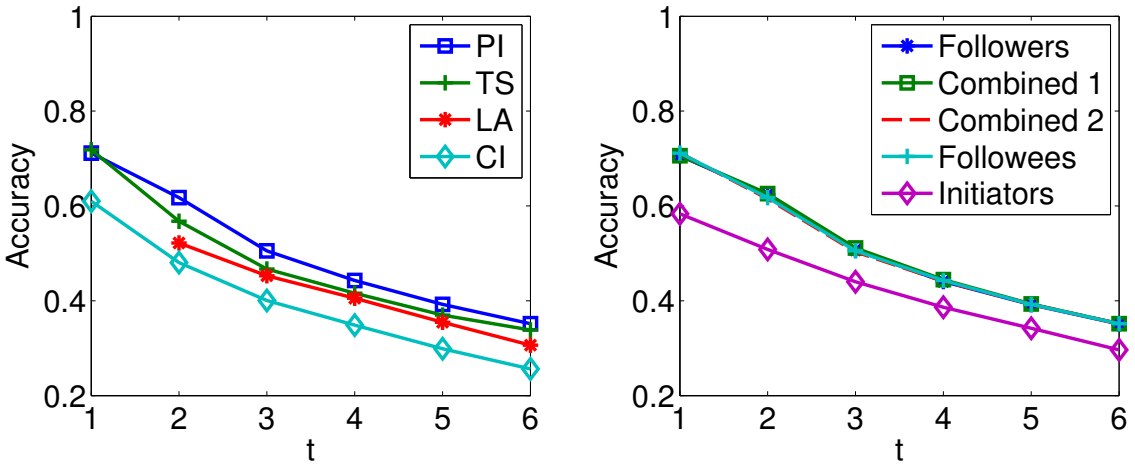
(a) TS (b) PI

Figure 4.4: The Impact of α.

bin, which is defined as the ratio of the number of positive ground-truth users in the i-th bin and the number of users in each bin and is computed as $100x_i/|U'|$. We can see that the accuracy in each bin decreases as the bin index increases, which is expected, as the later the users are added to U', the less likely they are indeed located in $\mathbb{A}$.

Fig. 4.4 shows the impact of $\alpha = |\overline{S}|/|S|$ on the accuracy of LocInfer. As expected, the accuracy under all locality metrics increases as α increases. The reason is that the larger the α, the more seeds, and the easier the target users in $\mathbb{A}$ can be discovered. The downside is that more seeds lead to a larger candidate set and thus higher crawling and computational cost, as Alg. 2 needs to check all the neighbors of the seeds.

Fig. 4.5 shows the impact of t on the accuracy by varying t from one to six. Specifically, Fig. 4.5a shows the accuracy for four areas using the followee locality, while Fig. 4.5b shows the accuracy for different locality metrics by using the PI dataset. Both figures show the accuracy decreases as t increases. This is expected



(a) For different locations.

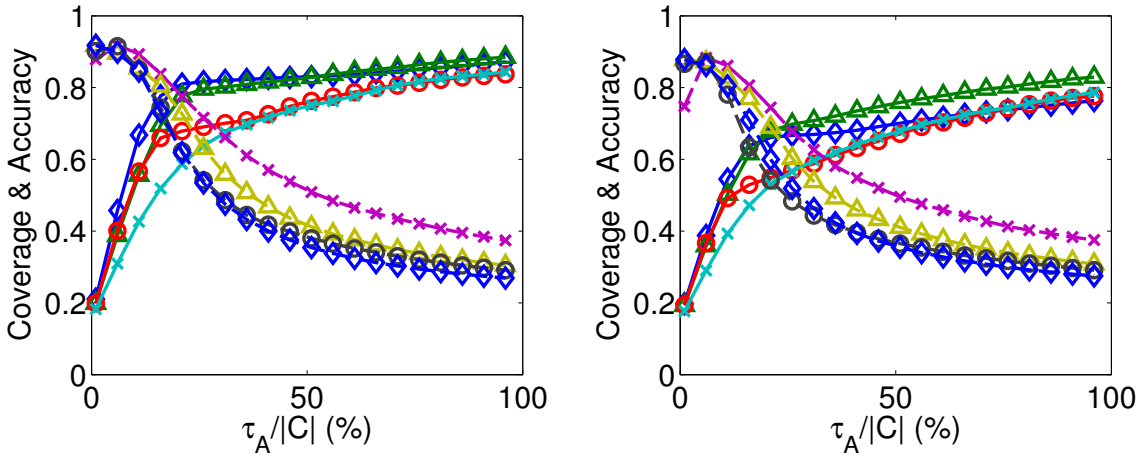
(b) For different localities.

Figure 4.5: The Impact of t.

because increasing t will result in the decrease in the size of candidate set and hence miss more users in the target user list who have no chance to appear in the candidate set. However, there is a tradeoff between the accuracy and cost because smaller candidate set will also bring the lower crawling and computational cost.

4.5.3 Coverage

Fig. 4.6 shows the coverage of LocInfer when $\alpha = 0.159$ with the desired number of target users (i.e., $\tau_{\mathbb{A}} = |U|$), varying from zero to the whole candidate set size $|C|$. We use both the followee and follower locality in this experiment. As expected, the larger $\tau_{\mathbb{A}}$, the more users in T contained in U, the higher coverage, and vice versa. When we set $\tau_{\mathbb{A}} = |C|$, the average coverage of these four locations by using followee, follower, and initiator locality is equal to 86.3%, 86.6%, and 79.7%, respectively. As stated, since the interacting edges (corresponding to replies, mentions, and retweets) are much sparser than following edges in the directed Twitter multigraph as shown in Fig. 4.1, the initiator locality has less coverage than the followee and follower



(a) Followees

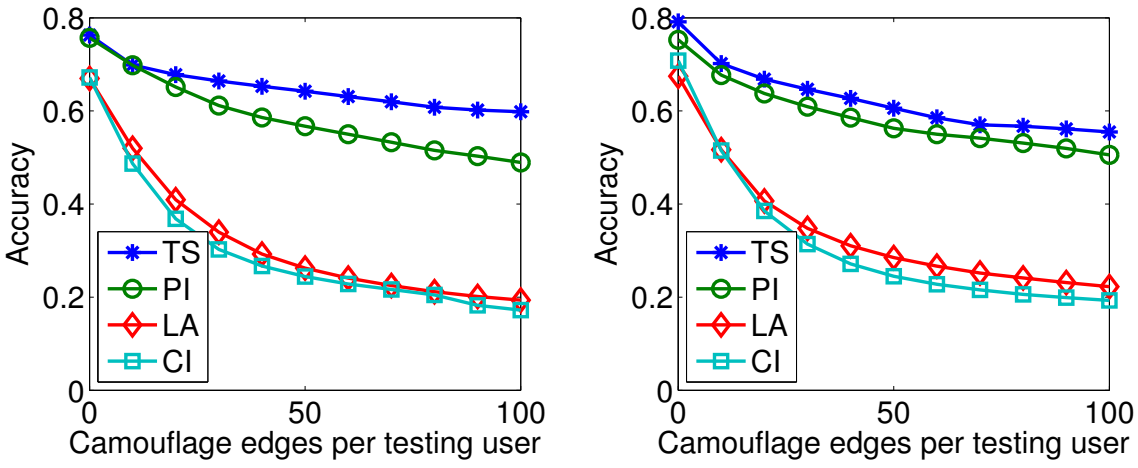
(b) Initiators

Figure 4.6: The Tradeoff Between the Coverage and Accuracy. The Solid and Dash Curves are the Coverage and Accuracy; the Marks $\diamond, \triangle, \circ, \times$ Represent `TS`, `PI`, `CI`, and `LA`, Respectively.

locality. Moreover, the average coverage by using the follower or followee locality in Fig. 4.6 is consistent with Corollary 4.4.1. Specifically, the average number of mutual followers d_m for four datasets is 7.8, 9.0, 11.6, and 11.6, respectively. According to Corollary 4.4.1, when $\alpha = 0.159$, $\overline{r}(t = 1) \geq 82.3\%$ which coincides with our results.

4.5.4 Accuracy and Coverage Tradeoff

Fig. 4.6 also shows the anticipated tradeoff between the coverage and accuracy. As we can see, the larger $\tau_{\mathbb{A}}$, the more the positive ground-truth users will be added to U, resulting in higher coverage. However, a larger $\tau_{\mathbb{A}}$ will also introduce negative ground-truth users into U, resulting in lower accuracy. This tradeoff could guide us to choose the parameter $\tau_{\mathbb{A}}$. On the one hand, if one desires higher coverage, a large termination threshold τ_A should be used, but it is possible that many users in U may be not indeed in $\mathbb{A}$. On the other hand, if one wants to be certain that the users



(a) Followees

(b) Followers

Figure 4.7: Countermeasure Efficacy.

discovered by LocInfer are most likely in $\mathbb{A}$, a smaller τ_A should be used at the cost of possibly missing some users indeed in $\mathbb{A}$.

4.5.5 *Effectiveness of Countermeasure*

To evaluate the efficacy of this countermeasure, we let each user in the testing set T in each area additionally follow or be followed by a certain number of users from Θ who are not in $\mathbb{A}$, and we refer to those following edges as *camouflage* edges. Fig. 4.7 shows the accuracy result under this countermeasure. As we can see, the accuracy of LocInfer decreases as the number of camouflage edges increases, highlighting the efficacy of the countermeasure. Besides adding random following edges, a user can also retweet, mention, and reply to random users on a regular basis to counteract LocInfer, which is expected to yield the similar results as these interactions can also decrease the geographic locality.

4.6 Related Work

In this section, we briefly present the existing work mostly related to this chapter.

Inferring a Twitter user's hidden location has been widely studied in the community, which can be categorized as contented-based and network-based methods. Content-based methods (65; 33; 96; 97) try to infer the user's location by his tweets. For example, Cheng *et al.* (33) proposed a probabilistic framework to estimate a Twitter user's location based on his tweets, resulting in placing 51% of Twitter users within 100 miles of their home locations. Mahmud *et al.* (97) further improved this result to 64% for city-level location inference. Hecht *et al.* (65) thoroughly studied the location profiles for the Twitter users and found that 34% of the users either left them empty or just non-geographic information. They also inferred the user's country and state information by checking their tweets. Network-based methods try to estimate a Twitter user's locations by his neighbors (15; 100; 73; 149; 38). Jurgens (73) aimed to infer all the users' location by building a global networks and then propagating location assignments from several seeds. Yamaguchi *et al.* (149) built several distributed landmarks and then inferred a user's location based on the connections with them. Compton *et al.* (38) inferred the locations of all the users in Twitter by minimizing their distances with the labelled users. Moreover, Li *et al.* (88) combined the content and network information to obtain the more accurate estimation. All these schemes seek to address the same question: how can we infer a user's hidden location from all his location-related tweets and/or neighbors' locations? This report targets a different problem: could we discover all or the majority of Twitter users in a metropolitan area? Directly adopting these existing methods to address our problem will result in scanning the whole Twitter network. Moreover, the accuracy of LocInfer outperforms the state of the art in (97).

This chapter is also related to privacy disclosure and protection in OSNs in general. Li *et al.* (87) used the neighbors' locations to infer the location in the emerging location-based social networks. Sun *et al.* (127) protected the location privacy on the social crowdsourcing networks. Mao *et al.* (98) used the tweets to detect the Twitter users' situational leak such as vacation status, drunk status, and medical conditions. Dey *et al.* (43) leveraged the information from neighbors to estimate the age of Facebook users. Mislove *et al.* (102) also used the local connections around the Facebook users to infer their hidden attributes such as major, college, and political view. Our work is complementary to these work and also highlights that current OSNs have emerged as an arguable threat to users' privacy.

4.7 Summary

This chapter presented LocInfer, a novel system that is able to discover the majority of Twitter users in any geographic area. Detailed experiments confirmed the high efficacy and efficiency of LocInfer. We also proposed a countermeasure to hide the locations of sensitive users from LocInfer and evaluated its efficacy with experiments driven by real datasets.

Chapter 5

YOUR AGE IS NO SECRET: INFERRING MICROBLOGGERS' AGES VIA CONTENT AND INTERACTION ANALYSIS

5.1 Introduction

Age information is much scarcer in microblogging systems than in traditional online social networks (OSNs). In a traditional OSN such as Facebook or LinkedIn, the users aim to maintain their personal identities and social connections with friends, so their personal profiles often contain true birthdate, school finishing/enrollment time, and other sensitive information, which can be directly used to infer accurate user ages (43). As more open social-networking platforms, however, microblogging systems are more informal than traditional OSNs in terms of maintaining social identities such that their user profiles often have no specific age-related information.

Age information in microblogging systems have important applications in both positive and negative ways. As an example for the positive aspect, the ability to select a group of users in the specific age range can enable numerous social and health studies such as investigating the diet habits of the college students between 18 and 22 years old and monitoring the workout habit of elderly or middle-aged people. The age information is also useful for cost-effective business marketing. For example, to launch a viral marketing campaign for a new wearable device via Twitter, a known strategy is for the marketer to seed the product with a few selected influential users from 30 to 50 years old who can potentially influence a disproportionately large number of others and also quickly trigger a cascade of influence (160). As an example for the negative aspect, being able to infer the age and other latent attributes based on the users'

public information such as tweets can help the adversary better profile the users for planning more advanced attacks such as spam campaigns or phishing attacks aiming at the elderly.

Accurate age inference is still an open challenge in microblogging systems due to three reasons. First, as stated before, the age information in microblogging systems is scarce. The microblogging service provider such as Twitter offers no explicit channel for users to indicate their age information. Therefore, existing methods that inferring a user's personal attributes directly from his/her online social neighbors' (102; 43; 90) are inapplicable because the neighbors' age information is also missing. Second, the microblogging messages (*microblogs* for short) posted by the users are highly unstructured, noisy, and massive. For example, each tweet in Twitter is composed of at most 140 characters, and hence microbloggers have created various slang and abbreviations to express their feelings and opinions, such as "wish4u a gr8 day" meaning "wish for you a great day." There are about 500M tweets per day, and each user is allowed to send up to 1000 tweets per day. These constrains make traditional text analysis for regular documents inapplicable in our context (67). Finally, the content information of the microbloggers is connected by online interactions such as following and retweeting. Traditional content analysis treating each user's content information independently fails to explore such rich online interactions.

In this chapter, we propose a new framework to infer microbloggers' ages by seamlessly integrating the content and interaction information on microblogging systems. Our key idea is driven by the presence of *homophily*, which has been discovered in many social studies (156). In our context, homophily refers to the tendency of a microblogger to associate and bond with similar others. For example, two colleague alumni in the same age group would be more inclined to follow each other in microblogging systems. In addition, the microbloggers with more intensive online inter-

actions are very likely to have more similar content information in their microblogs. To fully leverage the presence of homophily in microblogging systems, this chapter aims to answer two critical questions. First, how can we model both the content and interaction information in microblogging systems? Second, how can we effectively combine the content and interaction information together to accurately infer a microblogger's age?

Our contributions are summarized as follows.

- We motivate and formally define the age inference problem in microblogging systems with both content and interaction information.
- We propose MAIF, a unified framework to model and seamlessly integrate both the content and interaction information by considering the homophily of the content information among connected microbloggers.
- We thoroughly evaluate the proposed framework on a real-world dataset with 54,879 Twitter users, the largest in the community. Our results show that MAIF can achieve up to 81.38% inference accuracy and outperforms the state of the art by 9.15%.
- We outline some countermeasures for those wishing to preserve age privacy if our system were in place.

The rest of this chapter is organized as follows. Section 5.2 introduces the background and defines the problem. Section 5.3 details the age inference framework. Section 6.4 evaluates the proposed framework. Section 6.5 surveys the related work. Section 6.6 summarizes this chapter and future work.

5.2 Background and Problem Statement

In this chapter, we use Twitter as a representative microblogging system to illustrate our proposed framework. In what follows, we briefly introduce Twitter and then formally define the age inference problem.

After registering an account in Twitter, a user can post text-based microblogging messages of up to 140 characters, known as *tweet*s. S/he can also retweet, reply to, mark favorite any other public Twitter user's tweets. The user can also mention anyone else in the tweet by @someone. Unlike Facebook-like OSNs, the social relationships in Twitter are unidirectional by users *following* others. If user A follows user B, A is B's *follower*, and B is A's *followee*. In this chapter, we call A and B are *friends* if and only if A and B follow each other.

In this chapter, we are interested in classifying a user into one of c predefined age groups, which are further defined according to a widely-used adult development model (86) in Section 5.4.1. We do not want to infer the user's exact age for two main reasons. First, we observed that the majority of commercial advertisements and online surveys focus on the users of a specific age group. Hence our framework can well satisfy the requirements of such important applications. Second, there are no enough labeled users to infer exact ages. Nevertheless, our framework is flexible and extensible to infer the exact age by having one group per age as long as there are sufficient labeled users.

We assume that there is a set of *labeled* users in Twitter with explicit age information specified through tweets or other sources. As stated before, labeled users in microblogging systems are scarce. To tackle this challenge, we design a novel method to collect sufficient labeled users for building and evaluating our proposed framework. The details for labeled user collection are postponed to Section 5.3.1.

Problem Formulation. We formally model the microblogger's age inference problem as follows. Let $\mathcal{U}$ denote a set of n labeled users, $\mathcal{U}_{\mathcal{F}} \subseteq \mathcal{U}$ denote the union of each labeled user's friends in $\mathcal{U}$, $\mathcal{X}_u$ represent the microblogging messages of each user $u \in \mathcal{U}$ in the past year from the same given date, and $\mathbf{Y} \in \mathbb{R}^{n \times c}$ be an age-label matrix in which c is the number of classes, and $\mathbf{Y}_{i,j}$ is equal to 1 if user i is in age group j and 0 otherwise. We aim to build a classifier $\mathbf{W}$ to automatically assign the age labels for unknown users according to their microblogging messages. Here we leverage online interaction information (if there is) to train the classifier but do not need it for labeling unknown users, which is critical for the usability of the framework because the labeled users are scarce and so for the interactions between the unknown and labeled users.

5.3 Microbloggers' Age Inference Framework

As mentioned before, it is very challenging to infer the age information of Twitter users because of the tweets' unstructured, noisy, and massive nature as well as the scarcity of labeled users in Twitter. In this section, we first conduct an analysis of a dataset which is crawled via a novel method, and the analysis motivates the design of our microblogger's age inference framework (MAIF for short). Then we present a content metric to model each user u's tweet set $\mathcal{X}_u$ in Section 5.3.2. Next, we adopt a sparse representation method to model the content information for age inference in Section 5.3.3 and then use community structure to model the interaction information in Section 5.3.4. Finally, we integrate the content and interaction information to formulate the age inference problem as a convex optimization problem in Section 5.3.5 and then present our solution in Section 5.3.6.

5.3.1 Data Crawling and Analysis

We design a method to crawl the ground-truth labeled users. Inspired by (156; 90), we found that many users like to send their birthday greetings to their friends by posting a tweet containing two parts: a phrase of "happy yth birthday" where y is the age of the friend, and a mentioned user who is likely to be the friend's Twitter name. For example, user A has posted a tweet "Happy 24th Birthday to my best friend @B." It is clear that user B is 24 years old now. We then use Twitter's Streaming API to record all the tweets which contain one of the keywords "happy yth birthday" with y ranging from 14 to 70. Since the tweets are noisy, we use the following tricks to refine the collected tweets. First, we only select the tweets which mention only one person because it is very difficult to determine which user has the age information if more than one user have been mentioned. Moreover, if the tweet sender and the mentioned user are not friends, the tweet is excluded. This trick is to deal with the cases that the sender may just mention and require a celebrity to greet the sender's friend (e.g., an ordinary person, not mentioned). Since our framework relies on credible interactions among the users, such tweets and the corresponding users should not be considered. Finally, each user mentioned in the remaining tweets is assigned an age label y from the tweet, and we check the labeled users manually to exclude the users who are obviously not at the labeled age. The readers can check (77) for more details on how to crawl and analyze the Twitter system.

Based on the above method, we crawled the largest age-based ground-truth Twitter dataset in the community which is composed of 54,879 labeled users, each user's labeled friends, and each user's tweets from June 1, 2014 to May 30, 2015. Fig. 5.1 shows the age distribution of our dataset, which is consistent with the result in (156; 90). As we can see, 88.06% of labeled users are aged below 24, which is expected because

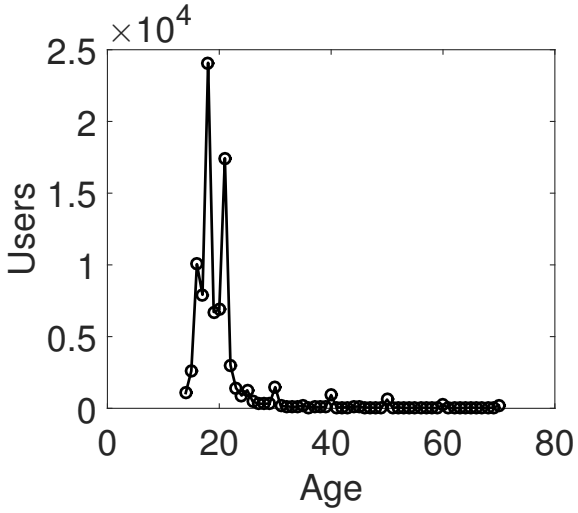
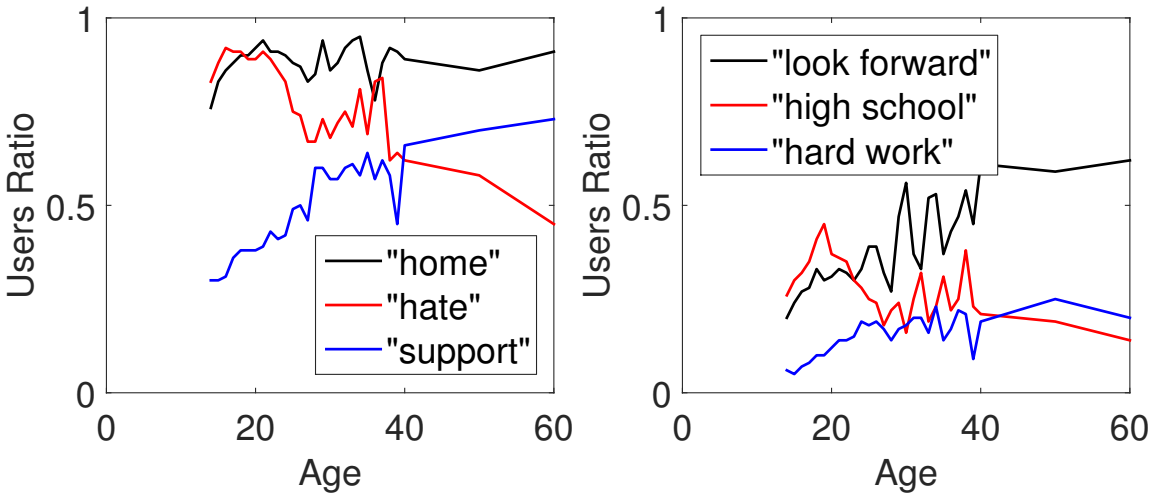


Figure 5.1: The Age Distribution in the Ground-truth Dataset.

young people are more likely to explicitly express their greetings using the social media. We notice that the dataset is biased toward young people, as Pew shows that 47% of Twitter users are older than 30 years old (46). However, the datasets with the similar distribution have been used in many previous work (156; 90), and it is still valuable and reliable for motivating our system design. We will also evaluate the impact of the biased dataset on system performance in Section 6.4.

Fig. 5.2 shows the generation gap (61) in terms of the word usage. Specifically, we selected six keywords, "home", "hate", "support", "look forward", "high school", "hard work", and check how many users at each specific age have used them in their tweet corpus. We can see that people with different ages have different keyword usage patterns. For example, users aged from 18 to 21 increase the usage of "home" because they might leave home for colleges; older people are less likely to use "hate" because they are more mature, but they are more likely to use "hard work" because they are highly engaged in the professional work; etc.

Fig. 5.3 and Fig. 5.4 demonstrate the social homophily in Twitter. Specifically, we first investigate the similarity in the ages of the users who have online interactions.



(a) Example 1

(b) Example 2

Figure 5.2: The Age-keyword Usage Pattern.

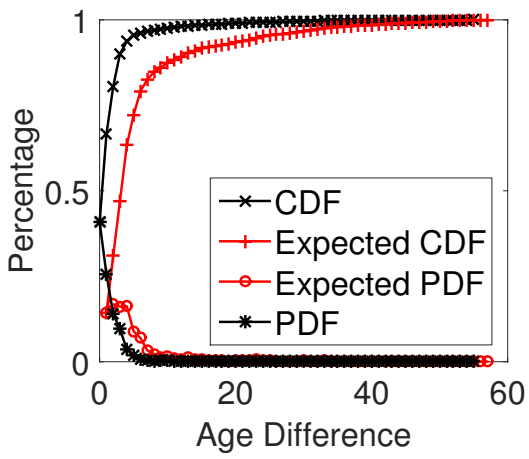


Figure 5.3: The Distribution of the Age Gap on Friend Pairs.

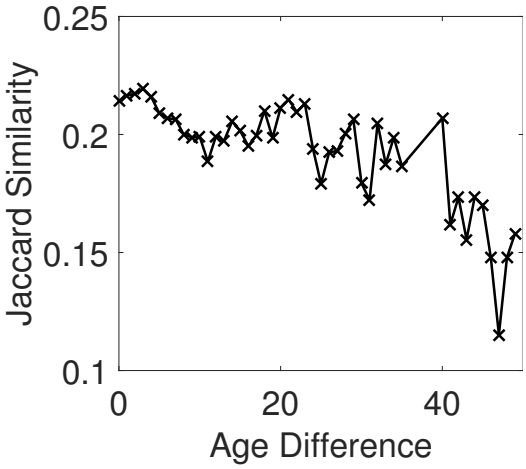


Figure 5.4: The Jaccard Content Similarity on Friend Pairs.

For this purpose, we measure the age difference of each friend pair in the dataset and draw the distribution in Fig. 5.3. To evaluate the impact of dataset bias, we also calculate the expected distribution of the age difference. To that end, we let each user befriend with each of other 54,878 users, and then measure the number of user pairs with a specific age difference. As we can see, in the original dataset, 40.84% of friend pairs have the same age, and 93.64% of the pairs have the age difference within 5 years while only 14.21% and 63.32% of the pairs in the fully-connected network have the same age and the age difference within 5 years, respectively. To measure the corpus similarity of each friend pair, we treat each user's tweets as a set of words and compute a Jaccard metric as $\frac{|\mathcal{A} \cap \mathcal{B}|}{|\mathcal{A} \cup \mathcal{B}|}$, where $\mathcal{A}$ and $\mathcal{B}$ denote the word sets for the two users involved, respectively. For all the friend pairs with the same age difference, we average their Jaccard similarities. As shown in Fig. 5.4, the corpus similarity decreases as the age difference of a friend pair increases.

We can draw two observations from the above analysis. First, the users at different ages have different topics in their tweets due to the age gap. Second, because of the social homophily (156), a user is more likely to befriend with thoses of the similar age, and their tweet topics tend to have higher similarity than the friend pairs with large age difference. These two observations drive us to design a framework to well integrate the content and interaction information to infer a Twitter user's age.

5.3.2 Model Tweets by τ-gram

Given the $\mathcal{X}_i$ of tweets of any labeled user $i \in \{1, \cdots, n\}$ in the past year, we first need to construct a mathematical model to represent it. Here we use a feature matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$ to model labeled users' tweets, where m refers to the dimension of a feature space $\mathcal{F}$ in the whole message space. In what follows, we describe how to construct the feature space $\mathcal{F}$ and then the feature matrix $\mathbf{X}$.

We first remove *stop words* in a stop-word list, [1] in which the words such as "the" and "those" are considered more general and meaningless. Then we conduct stemming (116) to reduce inflected words to their stem forms such that the words with different forms can be related to the same word. For example, "watch", "watching", and "watched" are all reduced to "watch".

Next, we represent the feature space for the cleansed tweets using a τ-gram technique, which is widely used for statistical text analysis. The τ-gram technique splits a give message into sequences of τ contiguous words, each referred to as a τ-gram with τ ranging from 1 to the message length. For example, consider a tweet {"Playing basketball against those guys was a bad idea"}. After removing stop words and performing stemming, we have {"play basketball against guy bad idea"}. The corresponding 1-grams are {"play", "basketball", "against", "guy", "bad", "idea"}, and the corresponding 2-grams are {"play basketball", "basketball against", "against guy", "guy bad", "bad idea"}. We let $\mathcal{N}_i$ denote the τ-grams of $\mathcal{X}_i$ for each user $i \in \mathcal{U}$ for all possible values of τ. Then we choose the top m most frequent τ-grams in $\bigcup_{1 \leq i \leq n} \mathcal{N}_i$ as the feature space $\mathcal{F}$.

Finally, we use the Term Frequency Inverse Document Frequency (TF-IDF) technique (85) to derive each element $\mathbf{X}_{i,j}$ in $\mathbf{X}$. Specifically, let $\Gamma(j)$ be the number of times a τ-gram j appears in the τ-gram list $\mathcal{N}_i$ of user i, $\Gamma_i^* = \max_{j \in \mathcal{N}_i} \Gamma(j)$, and $\Gamma'(j)$ denote the number of users in $\mathcal{U}$ whose τ-gram lists contain j. We define

$$\mathbf{X}_{i,j} = (0.5 + 0.5 * \frac{\Gamma(j)}{\Gamma_i^*}) * \log(\frac{n}{\Gamma'(j)}) \ . \tag{5.1}$$

The above normalization based on Γ_i^* is necessary because the users normally have very different tweet sets and thus different τ-gram lists. We refer interested readers to (85) for the details of the TF-IDF technique.

[1] `http://www.lextek.com/manuals/onix/`

It is a common practice to use 1-grams and 2-grams only for high computational efficiency without significantly sacrificing the analysis accuracy. So the feature space and matrix can be constructed very quickly in practice.

5.3.3 Modeling Content Information

Given the feature matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$ and the age-label matrix $\mathbf{Y} \in \mathbb{R}^{n \times c}$, a traditional method to build the classifier $\mathbf{W}$ is Least Square optimization(81), which learns a weighted model to minimize the estimation and the labeled data by solving

$$\min_{\mathbf{W}} \quad \frac{1}{2}\|\mathbf{XW} - \mathbf{Y}\|_F^2, \tag{5.2}$$

where $\|\mathbf{A}\|_F$ represent the Frobenius norm of matrix $\mathbf{A}$ which is defined as $\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m}\mathbf{A}_{i,j}^2}$.

The traditional Least Square method for a large feature set can lead to overfitting (134) in that the learned model may be too specific due to the limited training data and thus be inaccurate for inferring the ages of unknown users. Moreover, it has been observed in many domains that the underlying representations of many objects are sparse. For example, a signal could be efficiently reconstructed by far fewer samples in compressive sensing (17); when people speed-read documents, they may seek a sparse representation with key phrases or words instead of fully understanding every single word (99). These sparse features represent the given object more accurately and efficiently by capturing its underlying essence. In addition, by selecting a sparse and meaningful group of τ-grams rather than non-intuitive ones for each user, it could help sociologists, market planners and even the public to understand the behavior of the people in different age groups. To find and explore these sparse features in our feature space, we can improve the model defined in Eq. (5.2) by assigning higher weight to the most representative τ-grams. One widely-used method(134) is to introduce the

ℓ_1-norm regularization for the weight matrix $\mathbf{W}$ as follows,

$$\min_{\mathbf{W}} \quad \frac{1}{2}\|\mathbf{XW} - \mathbf{Y}\|_F^2 + \lambda_1\|\mathbf{W}\|_1, \tag{5.3}$$

where $\|\mathbf{W}\|_1 = \sum_{i=1}^{n}\sum_{j=1}^{m}|\mathbf{W}_{i,j}|$, and λ_1 is the parameter to control the sparse regularization. By adding this ℓ_1-norm constraint to the minimization problem, it enforces the coefficients of many non-representative features in $\mathbf{W}$ to be zero, thus making these features have no effect on the prediction model. With this strategy, we select relatively more "important" features (equivalently, τ-grams) to represent each age group.

5.3.4 Modelling Online Interaction Information

The content information in Twitter is networked. As shown in Section 5.3.1, people within the same age group have higher probability to share content similarity and also befriend with each other. For example, two college classmates follow each other on Twitter, often discuss final exam preparations for the same course, and/or cheer for the wins of their college sports teams. Given such observations, the content model in Eq. (5.3) should assign higher weights to similar τ-grams, such as "final" and "exam", so that the two users can be classified into the same age group with high probability. How to achieve this, however, is challenging because the two users very likely also tweet on different topics. Below we present how to model the online interactions among labeled users and then how to integrate the interaction information into the content model in Eq. (5.3).

We use the community concept to model the online interactions among the labeled users. For this purpose, it is worth noting that we can construct an undirected social graph from the labeled dataset, where each vertex corresponds to a labeled user, and an edge exists between two users if and only if they are friends (i.e., each

other's follower and followee). It has been widely reported that the users with the similar attributes such as ages would connect with each other more than the users with different attributes, hence forming a local community (102). The community structure can be inferred by maximizing the *modularity* (109), which is defined as follows.

Definition 5.3.1 (Modularity)**.** *Given an undirected graph $G = \langle \mathcal{U}, E \rangle$, where $|\mathcal{U}| = n$ is the user set, and $e_{ij} \in E$ equals 1 if users i and j are friends and equals 0 otherwise. Assume that G has been partitioned into k communities, and that each user belongs to one and only one community. The modularity of this partition is defined as*

$$Q = \frac{1}{2t} \sum_{i,j} (e_{ij} - \frac{d_i d_j}{2t}) \delta(C_i, C_j), \tag{5.4}$$

where $t = \frac{1}{2} \sum_{i,j} e_{ij}$ is the number of edges in G, $d_i = \sum_j e_{ij}$ is the degree of user i, C_i is the community containing user i, and the $\delta-$function $\delta(C_i, C_j)$ is 1 if $C_i = C_j$ and 0 otherwise.

The intuition behind the modularity is as follows. $\frac{d_i d_j}{2m}$ represents the expectation that any two users with degree d_i and d_j could form an edge in the graph. If they are connected (i.e., $e_{ij} = 1$) and are in the same community (i.e., $C_i = C_j$), they will contribute to the whole modularity Q. If they are not connected (i.e., $e_{ij} = 0$) but are in the same community (i.e., $C_i = C_j$), they will reduce the modularity Q. Finally, if they are in different communities (i.e., $C_i \neq C_j$), they have no impact on Q. Hence, the more edges in the same community, the higher its modularity.

Next, we present how to infer the community structure by maximizing the modularity. Let matrix $\mathbf{G} \in \mathbb{R}^{n \times n}$ represent the adjacent matrix for graph G where $\mathbf{G}_{i,j}$ equals 1 if $e_{ij} = 1$ and 0 otherwise. Let matrix $\mathbf{C} \in \mathbb{R}^{n \times k}$ represent a community partition for G where $\mathbf{C}_{i,j}$ is 1 if user i is in community j, and 0 otherwise. Note that

$\sum_j \mathbf{C}_{i,j} = 1$ since any user belongs to one and only one community. Then we could formulate the community partition problem as

$$\max_{\mathbf{C}} \quad Tr(\mathbf{CMC^T}), \quad s.t. \quad \mathbf{CC^T} = \mathbf{I} \tag{5.5}$$

where

$$\mathbf{M} = \mathbf{G} - \frac{\mathbf{d}\mathbf{d}^T}{2t} \tag{5.6}$$

where $\mathbf{d}$ is the degree vector for G, and $Tr(\mathbf{A}) = \sum_i \mathbf{A}_{i,i}$ represents the sum of the diagonal elements of $\mathbf{A}$. Since this problem is NP-hard (109), we resort to the widely used Louvain method (24) to obtain the approximation result.

After the community structure $\mathbf{C}$ is obtained, we expect that the users from the same community are in the same age group. Therefore we can use the community structure to improve our model in Eq. (5.3). To that end, inspired by (130), given $\hat{\mathbf{Y}}$ as the estimated age group labels for all the users in $\mathcal{U}$, we first compute the scatter of user pairs who are in the same community but have been estimated in either the same age group or two different age groups as:

$$\mathbf{S} \quad = \hat{\mathbf{Y}}^T \mathbf{F}\mathbf{F}^T \hat{\mathbf{Y}} \tag{5.7}$$

where $\mathbf{F}$ is the weighted community indicator matrix, which can be obtained from $\mathbf{C}$ as

$$\mathbf{F} = \mathbf{C}(\mathbf{C}\mathbf{C}^T)^{-\frac{1}{2}}, \tag{5.8}$$

where $\mathbf{F}_{ij}$ equals $\frac{1}{\sqrt{f_j}}$ if user i is in community C_j with f_j users and equals 0 otherwise.

It can be easily found that in Eq. (5.7), since $\hat{\mathbf{Y}}^T\hat{\mathbf{Y}}$ is a diagonal matrix with the (i,i)-th element equal to the number of users in the i-th age group, the (i,i)-th element of $\mathbf{S}$ measures how many user pairs in the i-th age group are in the same community, and the (i,j)-th $(i \neq j)$ element of $\mathbf{S}$ measures how many user pairs in the i-th age group and j-th age group are in the same community. Therefore, in order

to classify the users in the same community into the same age range, we just need to maximize the sum of (i, i)-th element in $\mathbf{S}$, i.e.,

$$\max_{\mathbf{W}} \quad Tr(\mathbf{S}). \tag{5.9}$$

Note that we ignore the user pairs who are in the same community but in different age groups because they violate the community structure.

5.3.5 Integrating Content and Interaction Information

Many existing methods on age estimation use either content or interaction information independently by assuming that these two pieces of information are unrelated. This assumption is not valid according to the intuition and also our data analysis in Section 5.3.1. So we propose to integrate both the content and interaction information into a unified model.

Particularly, since $\hat{\mathbf{Y}} = \mathbf{XW}$, Eq. (5.9) can be re-written as

$$\max_{\mathbf{W}} \quad Tr(\mathbf{W}^T\mathbf{X}^T\mathbf{F}^T\mathbf{FXW}). \tag{5.10}$$

By considering both the content information and interaction information, the age estimation problem defined in Eq. (5.3) could be reformulated as follows,

$$\min_{\mathbf{W}} \quad \frac{1}{2}\|\mathbf{XW} - \mathbf{Y}\|_F^2 + \lambda_1\|\mathbf{W}\|_1 - \frac{\lambda_2}{2}Tr(\mathbf{W}^T\mathbf{X}^T\mathbf{F}^T\mathbf{FXW}), \tag{5.11}$$

where λ_1 and λ_2 are the parameters for sparse regularization (for content information) and integration of interaction information, respectively. By varying these two parameters, we could set the importance of sparse regularization and interaction integration on the original Least Square model.

5.3.6 An Optimization Algorithm

The problem defined in Eq. (5.11) is non-smooth because the ℓ_1 regularization $\|\mathbf{W}\|_1$ is not differentiable. Hence we transform it into its differentiable Lagrange

dual function as:

$$\begin{aligned} \min_{\mathbf{W}} \ & \frac{1}{2}\|\mathbf{XW}-\mathbf{Y}\|_F^2 - \frac{\lambda_2}{2}Tr(\mathbf{W}^T\mathbf{X}^T\mathbf{F}^T\mathbf{FXW}), \\ \text{s.t.} \ & \|\mathbf{W}\|_1 \leq z, \end{aligned} \tag{5.12}$$

where $z \geq 0$ is the radius of the ℓ_1-ball and has a one-to-one correspondence with λ_1. Let

$$f(\mathbf{W}) = \frac{1}{2}\|\mathbf{XW}-\mathbf{Y}\|_F^2 - \frac{\lambda_2}{2}Tr(\mathbf{W}^T\mathbf{X}^T\mathbf{F}^T\mathbf{FXW}), \tag{5.13}$$

we can see that $f(\mathbf{W})$ is a smooth objective function, and the optimization problem is convex which can be solved by gradient descending methods. It is known (22) that the gradient step

$$\mathbf{W}^{(k)} = \mathbf{W}^{(k-1)} - \frac{1}{t^{(k)}} \bigtriangledown f(\mathbf{W}^{(k-1)}) \tag{5.14}$$

for solving the smooth optimization problem in Eq. (5.12) can be treated as finding the minimum Euclidean projection (28) of $\mathbf{W}^{(k)}$ defined above on the ℓ_1-ball $\|\mathbf{W}\|_1 \leq z$, which is

$$\mathbf{W}^{(k)} = \arg\min_{\mathbf{W}} M_{t^{(k)}}(\mathbf{W}, \mathbf{W}^{(k-1)}), \tag{5.15}$$

$$\begin{aligned} M_{t^{(k)}}(\mathbf{W}, \mathbf{W}^{(k-1)}) =& f(\mathbf{W}) + \langle \mathbf{W}-\mathbf{W}^{(k-1)}, \bigtriangledown f(\mathbf{W}^{(k-1)})\rangle \\ & + \frac{t^{(k)}}{2}\|\mathbf{W}-\mathbf{W}^{(k-1)}\|_F^2, \end{aligned} \tag{5.16}$$

where $t^{(k)}$ is the step size, $\langle \mathbf{A}, \mathbf{B}\rangle = Tr(\mathbf{A}^T\mathbf{B})$ denotes the matrix inner product, and

$$\bigtriangledown f(\mathbf{W}^{(k-1)}) = \mathbf{X}^T\mathbf{XW}^{(k-1)} - \mathbf{X}^T\mathbf{Y} - \lambda_2\mathbf{X}^T\mathbf{F}^T\mathbf{FXW}^{(k-1)}. \tag{5.17}$$

Let $\mathbf{U}^{(k-1)} = \mathbf{W}^{(k-1)} - \frac{1}{t^{(k)}} \bigtriangledown f(\mathbf{W}^{(k-1)})$. The Euclidean projection in Eq. (5.15) has a closed-form solution (93) as follows,

$$\mathbf{w}_j^{(k)} = \begin{cases} (1 - \frac{\lambda_1}{t^{(k)}\|\mathbf{u}_j^{(k-1)}\|})\mathbf{u}_j^{(k-1)} & \text{if } \|\mathbf{u}_j^{(k-1)}\| \geq \frac{\lambda_1}{t^{(k)}} \\ 0 & \text{o.w.} \end{cases} \tag{5.18}$$

where $\mathbf{w}_j^{(k)}$ and $\mathbf{u}_j^{(k-1)}$ are the j-th rows of $\mathbf{W}^{(k)}$ and $\mathbf{U}^{(k-1)}$, respectively.

Algorithm 4 details the algorithm which comprises an outer loop and an inner loop. The inter loop from Line 4 to 9 searches the step size $t^{(k)}$ to solve the gradient step defined in Eq. (5.15) according to Eq. (5.18). The outer loop then updates the $\mathbf{W}^{(k)}$. To accelerate the gradient descent in Eq. (5.15), we build a linear combination of $\mathbf{W}^{(k)}$ and $\mathbf{W}^{(k-1)}$ as $\mathbf{H}^{(k)}$ in line 3 (70). The algorithm terminates when $|f(\mathbf{W}^{(k)}) - f(\mathbf{W}^{(k-1)})| \leq \epsilon |f(\mathbf{W}^{(k-1)})|$. Similar to the proof in (93), given the termination parameter ϵ, it is easy to verify that the convergence rate of our algorithm is $O(\frac{1}{\sqrt{\epsilon}})$.

Algorithm 4: Classier Training for Age Inference

input : $\mathbf{X}, \mathbf{Y}, \mathbf{F}, \lambda_1, \lambda_2, \epsilon$

output: $\mathbf{W}$, i.e., the feature-to-label matrix.

1 Initialize $\mathbf{W}^{(k)} \leftarrow \mathbf{0}$, $\eta^{(0)} \leftarrow 0$, $\eta^{(1)} \leftarrow 1$, $k \leftarrow 1$;

2 **while** $|f(\mathbf{W}^{(k)}) - f(\mathbf{W}^{(k-1)})| > \epsilon |f(\mathbf{W}^{(k-1)})|$ **do**

3 Set $\mathbf{H}^{(k)} \leftarrow \mathbf{W}^{(k)} + \frac{\eta^{(k-1)}-1}{\eta^{(k)}}(\mathbf{W}^{(k)} - \mathbf{W}^{(k-1)})$;

4 **while** *True* **do**

5 Set $\mathbf{U}^{(k-1)} \leftarrow \mathbf{H}^{(k-1)} - \frac{1}{t^{(k)}} \bigtriangledown f(\mathbf{W}^{(k-1)})$;

6 Compute $\mathbf{w}_j^{(k)}$ according to Eq. (5.18) ;

7 **if** $f(\mathbf{W}^{(k)}) \leq M_{t^{(k)}}(\mathbf{H}^{(k-1)}, \mathbf{W}^{(k)})$ **then**

8 break;

9 $t^{(k)} \leftarrow 2 \times t^{(k-1)}$;

10 $\mathbf{W} \leftarrow \mathbf{W}^{(k)}, \eta^{(k)} \leftarrow \frac{1+\sqrt{1+4(\eta^{(k-1)})^2}}{2}, k \leftarrow k+1$;

11 **return** $\mathbf{W}$.

5.3.7 *Inferring Age Group of an Unknown User*

After we build a classifier $\mathbf{W}$, we can estimate the age range of any unknown user u as follows. We crawl the tweets from u as $\mathcal{X}_u$ in the past year and then build the τ-gram list $\mathcal{N}_u$. Based on the feature space $\mathcal{F}$, we then construct the feature vector $\mathbf{x}_u \in \mathbb{R}^{1\times m}$ by calculating the TF-IDF of each τ-gram in $\mathcal{F}$ according to Eq. (6.1). The final step is to estimate the age group with the maximum likelihood as follows,

$$\arg\max_{i=\{1,2,\ldots,c\}} \quad \mathbf{x}_u\mathbf{w}_i, \tag{5.19}$$

where c is the number of age groups, and $\mathbf{w}_i \in \mathbb{R}^{m\times 1}$ is the i-th column of the classifier matrix $\mathbf{W}$. Note that this step needs no interaction information from user u. This feature can be very useful because it makes our algorithm above directly applicable to an arbitrary unknown user with or without interactions with labeled users in the classifier $\mathbf{W}$.

Note that MAIF needs the labelled users and their content/network information to build the classifier $\mathbf{W}$, which can be crawled by the method presented in Section 5.3.1. Due to the scarcity of the age information, the network information between the labelled users might be limited. However, MAIF could work even with zero network information, and as shown in the evaluation below, the richer the network information, the better the performance.

5.4 Evaluation

In this section, we thoroughly evaluate the proposed framework. Specifically, we want to answer these four questions:

1. How accurate is the proposed framework in comparison with other age inference schemes?

2. What is the impact of dataset bias on the performance?

3. What is the benefit of integrating both the content and social interaction information?

4. What is the impact of key parameters in the framework?

In what follows, we first introduce the dataset as well as the evaluation methodology and metrics. Then we seek to answer the above questions. Finally, we briefly discuss possible countermeasures for sensitive users to preserve their age privacy if our framework were deployed.

5.4.1 Dataset, Methodology and Metrics

We first partition the Twitter users into five groups according to Levinson's adult development model (86):

- Group 1: 14-18. This group is for juvenile and adolescence users. Since Twitter only allows the users older than 13 years to access the service, we start this group from 14 years old.

- Group 2: 19-22. According to Levinson's model, this group is a transition phase from the pre-adulthood to the early adulthood. People in this age group are usually enrolled in the college.

- Group 3: 23-33. This group is the "time for building and maintaining an initial mode of adult living." People within this age group are beginning their professional career, building the family, or getting prepared for their career by further graduate study.

- Group 4: 34-45. This is the phase of early adulthood to define a new era which belongs to them.

- Group 5: > 46. This group include people from 46 to 65 who are in their middle adulthood and people who are older than 65 in the phase of the late adulthood.

We use two datasets to evaluate the proposed framework, as shown in Table 5.1. First, the original dataset crawled in Section 5.3.1 is partitioned to five age groups as described above. Fig. 5.1 shows that the age distribution is highly biased toward Group 1 and 2, which occupy 88.06% of all the users. This is because the young people are more active in posting their birthday greetings to their friends. We first evaluate MAIF on this original dataset. Moreover, to evaluate the impact of the dataset bias, we build a comparable and balanced dataset as follows. We keep all the users in Group 3, which have 2,986 users, and then randomly sample the same number of users from both Group 1 and 2. After sampling, the network is less connected. Specifically, in the original dataset, each user has on average 1.06 friends within the dataset in contrast to 0.141 friends in the sampled dataset.

We use cross validation to evaluate the proposed framework. Specifically, given a ground-truth dataset composed of users who have indicated their ages, we split it into five subsets and conducted the experiment by five rounds. In each round, we choose four different subsets to build the classifier $\mathbf{W}$, then apply it to the remaining subset to estimate the users' ages, and finally compare them with the ground truth.

Since we aim to classify a user into $c (c > 2)$ groups, we derive both the *separate accuracy* for each group and the *overall accuracy* for all the groups from the *confusion matrix* [2] . For each age group i, we denote the number of true positives, false positives, true negatives, and false negatives by $\#\mathtt{TP}_i, \#\mathtt{FP}_i, \#\mathtt{TN}_i$, and $\#\mathtt{FN}_i$, respectively. Then we define the $\mathtt{Precision}_i, \mathtt{Recall}_i, \mathtt{F-score}_i$ as the separate accuracy for age group

[2]Here we didn't use the confusion matrix directly because it is not efficient to compare the MAIF with several baseline methods. However, the derived separate and overall accuracy can represent well the confusion matrix.

Table 5.1: The Summary of the Datasets.

Datasets	#Users	#Age Groups	Age Group Dist.	#Tweets	#Edges (Avg.)	k
Original	54,879	Group 1-5	[0.505, 0.376, 0.076, 0.022, 0.021]	51,756,652	58,267 (1.06)	19,978
Sampled	8,958	Group 1-3	[0.333, 0.333, 0.333]	8,567,085	1,263 (0.141)	7,743

i as follows:

$$\begin{aligned} \texttt{Precision}_i &= \frac{\#\texttt{TP}_i}{\#\texttt{TP}_i + \#\texttt{FP}_i}; \quad \texttt{Recall}_i = \frac{\#\texttt{TP}_i}{\#\texttt{TP}_i + \#\texttt{TN}_i}; \\ \texttt{F}-\texttt{Score}_i &= \frac{2 \times \texttt{Precision}_i \times \texttt{Recall}_i}{\texttt{Precision}_i + \texttt{Recall}_i}. \end{aligned} \tag{5.20}$$

We then define the overall accuracy as $\texttt{X} = \sum_{i=1}^{c} r_i \texttt{X}_i$, where $\texttt{X}$ represents $\texttt{Precision}, \texttt{Recall}$, or $\texttt{F}-\texttt{score}$, and r_i is the ratio of users in age group i over the whole dataset, which is listed as the age distribution in Table 5.1.

5.4.2 Assessing Accuracy

We first evaluate the accuracy of the proposed framework and compare it with both the state-of-the-art methods and the baseline methods summarized as follows.

- Content-based methods I. The state-of-the-art content-based method is proposed by (110) to use the linear regression model with the ℓ_2 regularization, which is equivalent to adding $\|\mathbf{W}\|_F$ to the least square method defined in Eq. (5.2). We use the top-10000 1-gram and 2-gram as the features to infer the age information.

- Content-based methods II with sparse representation. We use the least square method with the ℓ_1 regularization in Eq. (5.3) to evaluate the inference performance by including the sparse representation for the content information.

- Neighbor-based method I. We infer the age information from neighbors' content information as used in (156; 31). Specifically, for each user i, we use the least square method with the ℓ_1 regularization in Eq. (5.3) to estimate the age information of i's f friends who are not in the labeled user set, and then set the average value as i's age information. In the experiment, we set f be 10 and 20.

- Neighbor-based method II. We infer the age information from labeled neighbors' age information as used in (43; 90). Specifically, we implement the more advanced method in (90) which assigns a weight between every friend pair in the labeled user set and then uses the label propagation to estimate the unknown users' ages. We use 80% of the users as the training set and the remaining as the testing set.

- The proposed framework. We set both λ_1 and λ_2 in the Eq. (5.11) to be 1 for the general experiment, and we will explore the effects of parameters later. Moreover, we set the size of the feature space $m = 10,000$ with 5,000 of 1-grams and 2-grams each and the termination condition $\epsilon = 10^{-4}$ in Alg. 4.

For each method, we compare the separate accuracy of each group and the overall accuracy, as shown in Table 5.2. We could draw three conclusions from the overall accuracy. First, the proposed MAIF is better than all other four methods, verifying that our framework can accurately integrate the content information and the interaction information, which are the essential behavior pattern of twitterers, to infer the age information. Second, the sparse representation in Content-II method outperforms the least square in Content-I, meaning that the content information in microblogging services is indeed sparse, and that the sparse features could represent age groups more accurately. Third, directly inferring the age information from labeled neighbors' age information as in Neighbor-II method is not effective for the dataset. The reason is that the age information in Twitter is so scarce that many users lack the neighbors who have specified their ages. As we can see from Table 5.1, the average friends in the original dataset is 1.06, meaning that every labeled user only has average one friend in the dataset. To overcome this issue, MAIF leverages the community structure which contains more users and integrates it with the content information.

Table 5.2: The Performance on the Original Dataset.

	Average Accuracy			F-score for each group				
	Precision	Recall	F-score	Group 1	2	3	4	5
Content-I	0.7397	0.7538	0.7456	0.8246	0.7300	0.2451	0.1022	0.0301
Content-II	0.7435	0.7585	0.7495	0.8284	0.7349	0.2481	0.0670	0.0465
Neighbor-I ($f = 10$)	0.6677	0.6816	0.6557	0.7737	0.5883	0.0694	0.3281	0.1429
Neighbor-I ($f = 20$)	0.6886	0.7038	0.6809	0.7879	0.6318	0.0952	0.2642	0
Neighbor-II	0.4861	0.5021	0.4899	0.5589	0.4729	0	0	0
MAIF	**0.8069**	**0.8349**	**0.8138**	**0.9022**	**0.8196**	0.1795	0.0122	0.0441

Table 5.3: The Performance on the Sampled Dataset.

	Average Accuracy			F-score for each group		
	Precision	Recall	F-score	Group 1	Group 2	Group 3
Content-I	0.5828	0.5829	0.5823	0.5661	0.5253	0.6557
Content-II	0.6930	0.6946	0.6935	0.6857	0.6221	0.7726
Neighbor-I ($f = 10$)	0.5606	0.5626	0.5073	0.6547	0.2125	0.6548
Neighbor-I ($f = 20$)	0.5721	0.5663	0.5078	0.6824	0.1930	0.6481
Neighbor-II	0.2392	0.3460	0.2506	0.2808	0.4787	0
MAIF	**0.7587**	**0.7611**	**0.7582**	**0.7572**	**0.6828**	**0.8347**

As for the separate accuracy, we could see that although MAIF outperforms other methods in Group 1 and 2, all methods have low accuracy for the remaining groups. We conjectured that the low accuracy for Group 3 to 5 is caused by the bias of the dataset. To verify this conjecture, we applied these five methods on the sampled dataset described in Table 5.1 and obtained the results in Table 5.3. The results show that MAIF outperforms all other four methods in both average and separate accuracy. Moreover, the accuracy of MAIF for each age group on this dataset is significantly balanced than on the original dataset, which justifies our conjecture and also answers the second question stated in the beginning of this section.

5.4.3 Performance of the Content and Interaction information

Since the proposed MAIF framework explores content and also interaction information, we aim to investigate the contribution of each type and the benefit of the integration. Specifically, we consider the following methods.

- Content-only methods. We use both the widely-used Support Vector Machine (SVM) (128) and the least square with the ℓ_1 regularization in Eq. (5.3) to evaluate the performance on the content feature matrix $\mathbf{X}$.
- Network-only methods. We use the adjacent matrix $\mathbf{G}$ as the feature matrix and then apply both SVM and the least square with the ℓ_1 regularization in Eq. (5.3) on them.

Fig. 5.5 shows the overall accuracy of content-only methods, network-only methods, and the proposed framework. As we can see, MAIF outperforms both content-only and network-only methods, meaning that the accuracy will increase if we integrate both content and interaction information instead of considering only one type of information. Moreover, content-only methods perform better than network-only

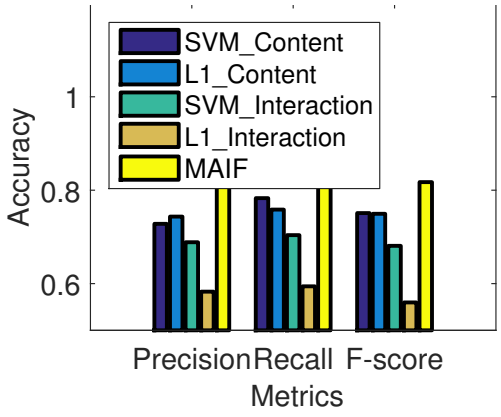


Figure 5.5: The Performance of Separate Information.

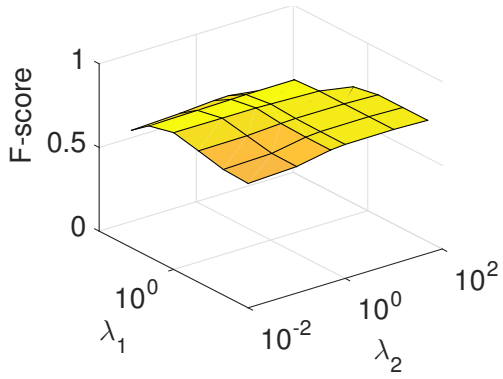


Figure 5.6: The Impact of the Parameters λ_1 and λ_2.

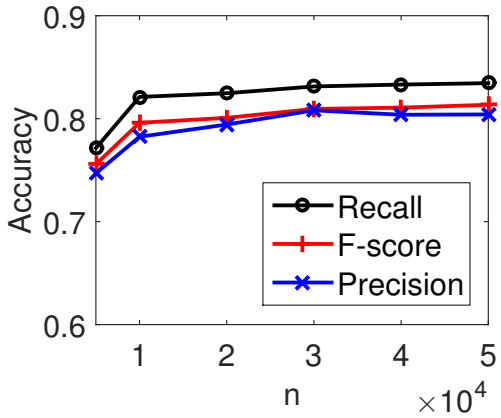


Figure 5.7: The Accuracy Under Different Dataset Sizes.

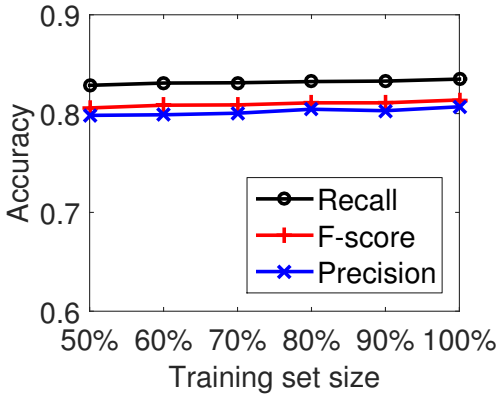


Figure 5.8: The Accuracy Under Different Training Set Sizes.

methods, indicating that content information is more reliable and contributes more than the network information to infer the age information. Again, we conjecture that this is caused by the scarcity of the age information in Twitter and hence the scarcity of the network information between the labeled users.

5.4.4 *Exploiting the Parameters*

We show the impact of λ_1 and λ_2 in Fig. 5.6. As stated before, the parameter λ_1 indicates the weight of the sparse representation of the content information, and the parameter λ_2 indicates the weight of interaction information. As we can see, both parameters lead to smooth and meaningful results when they are between 0.01 and 100. Moreover, when λ_2 increases from 0.01 to 100, the overall accuracy first increases and then decreases; similar trend holds for λ_2. Hence we expect a local optimal parameter pair for λ_1 and λ_2 to be both 1. Moreover, the accuracy in this range is smooth and has limited variance, suggesting that in practice we could choose these two parameters from 0.01 to 100.

Fig. 5.7 show the impact of the dataset size. Given the original dataset with 54,879 users, we randomly sample 5000, 10000, 20000, 30000, 40000, and 50000 users, respectively, and use five-fold cross validation to yield the results. As we can see, increasing the dataset size improves the accuracy because of the richer content and network information. Specifically, the more users, the richer content and network information which could better represent the users in a specific age group, and hence the higher accuracy.

Our algorithm is stable in terms of other parameters. Fig. 5.8 demonstrate the impact of the training set size. 50% of the training set means that only 50% of the 80%, which equals 40%, of the users in the whole dataset are used as the training set to predict the remaining 20% of users. The training set size varies from 50% to 100% (corresponds to 40% to 80% of all users). As expected, increasing the training set can slightly improve the accuracy. However, the improvement is less significant than increasing the dataset size, as shown in Fig. 5.7. The reason is that the training set, which has 40% of the 54,879 users, is still large enough to obtain the good results

even if we only use 50% of the training set. Moreover, we have evaluated the impact of feature space size m (from 5000 to 20000), the different combinations of 1-gram and 2-gram in the feature space, and the variant definitions of TF-IDF in Eq. (6.1), and obtained similar results, which are omitted here due to space constraints.

5.4.5 Countermeasures

The above experiments demonstrate the efficacy of using the public content and interaction information to infer the age information, one type of highly-private personal attributes. Since the social homophily and generation gap always exist, it is very challenging for twitterers to evade the inference demonstrated in this chapter. In order to preserve their privacy, one way is to set "protected"—a function provided by Twitter—the critical information such as followers, followees, and even all the tweets, such that only authorized users could visit while the unauthorized third party will fail to infer due to the absence of content and interaction information. Another way is to diversify the content and/or interaction information by posting with the style from other age groups and/or following people with different ages. As shown in Fig. 5.5, the absence of one type of information will lower the inference performance, so the age privacy can be protected to some extent. Nevertheless, this chapter mainly aims to demonstrate a more effective method to infer the hidden age information from public content and interaction information in Twitter. More privacy implications of our framework and the thorough investigation of countermeasures are beyond the scope of this chapter.

5.5 Related Work

In this section, we briefly present the existing work mostly related to this chapter. There has been some effort to infer hidden age information in microblogging sys-

tems. Nguyen *et al.* tried to classify the user ages from different angles such as age range, exact age, and life stage with the 1-grams constructed from the tweets (110). Oktay *et al.* proposed a method to infer users' age range by investigating their names. The idea is that different generations have different preferences on the baby naming (112). Liao *et al.* use the ages of online neighbors to infer the age of a given user (90). Dey *et al.* also used the similar method to infer the user age in Facebook (43). However, this method requires that some neighbors have specified their ages, which cannot be satisfied in microblogging systems where age information is scarce.

Other hidden attributes such as location (88; 97; 38; 159), gender (118), political preference (156), and ethnicity (31) have also been inferred by either the content information and/or the interaction information. Mislove *et al.* used the local connections around the Facebook users to infer their major, college, and political view (102). Location information has attracted many attentions recently. The content with geographical hints could be used to infer users' locations (97). Since about 16% of Twitter users have specified their locations, inferring users' locations from their neighbors' locations can be more effective (88; 38) than inferring their ages from their neighbors' ages. Besides the privacy threats by inferring these sensitive attributes, there have been growing security issues against social network users (26; 161; 162).

5.6 Summary

In this chapter, we propose MAIF, a novel framework which explores public content and interaction information in microblogging systems to infer the hidden ages of microbloggers. We thoroughly evaluate MAIF using a real-world dataset with 54,879 Twitter users. Our results show that MAIF can achieve up to 81.38% inference accuracy and outperforms the state of the art by 9.15%.

Chapter 6

PRIVACY PRESERVING SOCIAL MEDIA PUBLISHING

6.1 Introduction

User-generated social media data are exploding. Twitter, the most popular microblogging service, generates 500 million tweets per day by 320 million monthly active users as of December 2015. Facebook, the largest online social network with about 1 billion daily active users as of June 2015, generates 4 new petabytes of data per day. People use social media platforms to communicate with their friends, share their daily life experiences, express their opinions on political/social events and commercial products, etc.

Closely tied to human beings in the physical world, large-scale social media data have tremendous usages by various *data consumers* and have become one of the most profitable resources for social media service providers (55). For example, companies use the data to learn the behavior of their customers, monitor the public responses to their products, deliver online advertisements more cost-effectively, and uncover the trends that may impact their businesses; public policy makers explore the data to obtain the demographic information for making strategic decisions; public health agencies analyze the data to identify and predict disease outbreaks; and sociologists leverage the data to study the social behavior and establish new social network theories. In a typical social media mining application, the data consumer demands a set of users and their social media data (such as profiles, recent posts, and friends), which satisfy some desirable criterion. For example, company A may request the data of all the users who mentioned the company in the past week after a public relation crisis.

The disclosure of complete and intact social media data exacerbates the threats to user privacy. For example, many users mention their vacation plans in publicly visible tweets without knowing that criminals can exploit such information for targeted break-ins and thefts (98). Criminals may identify potential victims nearby by directly browsing/searching social media platforms, and smarter ones can exploring the search APIs offered by social media platforms. The data acquired in this traditional way are only small and random samples of all the qualifying data. For example, Twitter claims that their Search API only "searches against a sampling of recent tweets published in the past 7 days" (137). If the criminals could access the complete and intact social media data in the target area, they can identify all potential victims to plan large-scale break-ins. In addition, social media mining applications are increasingly sophisticated and powerful. If complete and intact social media data are available, lots of sensitive information the users do not explicitly disclose could still be inferred, such as age (156; 158), location (88; 159), language (110), and political preferences (31).

There is a natural conflict between data utility and user privacy in social media data publishing. On the one hand, data consumers want complete and intact social media data to maximize the data utility, which is also the most profitable case to social media service providers. The maximum data utility is achieved unfortunately at the biggest sacrifice of user privacy. On the other hand, social media service providers are also motivated to protect user privacy due to legal concerns, public relations, and many other reasons. For example, they may intentionally add random noise to the data before releasing them to data consumers. User privacy is thus better protected but at the loss of data utility.

A growing body of work studies privacy-preserving publishing of social graphs and falls into two directions. The first line of research (64; 94; 129) aims at vertex

privacy by publishing social graphs with anonymous user IDs, and the research effort is to prevent the adversary from linking anonymous IDs to corresponding users in the real social network. The other line of research targets link privacy (103; 92), and the main effort is to publish social graphs with real user IDs but perturbed links by deleting some real edges and adding some fake ones. Neither line of work considers the privacy of user data and thus cannot be directly applied in our context.

In this chapter, we propose a framework for privacy-preserving social media data publishing. The framework consists of a data service provider (DSP), numerous social media users, and a lot of data consumers. The DSP can be either a social media service provider itself such as Twitter or Facebook, or a third-party data company such as Gnip and DataSift which resells the data obtained from social media service providers. Data consumers can be an arbitrary individual or entity in public or private sectors. They are interested in statistical information that can be mined from social media data, rather than real user IDs. A data consumer submits a data request to the DSP, which specifies the query conditions. The DSP responds with a data set satisfying the query conditions, in which each user ID is anonymized.

Although there can be various attacks on social media data publishing, we consider a *user-linkage attack* as the first effort along this line. In this attack, a malicious data consumer attempts to link random or selected anonymous IDs in the received data set to real IDs on the social media platform, so he can obtain the latest social media data about the victims or other sensitive information not covered by his previous query. We assume that existing sophisticated techniques such as (64; 94; 129; 103; 92) are adopted to preserve both link privacy and vertex privacy in the anonymized data set, so the attacker cannot uncover real IDs based on either vertexes or edges. Our focus is to perturb user-generated data such that they cannot be exploited for the user-linkage attack.

It is very challenging to design effective defenses against the user-linkage attack while achieving high data utility. First, social media data are highly unstructured and noisy. Specifically, unlike traditional documental content, social media data are composed of a large mount of short and informal posts. For example, each tweet allows at most 140 characters. So users tend to use slang and abbreviations to express their feelings and opinions, such as "u have a gr8 day" meaning "you have a great day." This phenomenon makes traditional privacy-preserving techniques on structured databases inefficient (54). Second, social media data are extremely large in quantity and contain a lot of redundancy. For example, many users have intensive interactions with their friends in the forms or replies and retweets, or they may post many tweets on a single topic. Finally, different data consumers may have diverse social media mining applications with totally different utility functions, so the data they request from the DSP may differ significantly in variety and/or quantity. So it is almost impossible for the DSP to adopt a universal mechanism to preserve the same level of user privacy while satisfying the utility requirements of various data consumers.

Our defense against the user-linkage attack in this chapter consists of three steps. First, we map the complete and intact data of all the users into a high-dimensional user-keyword matrix. Second, we add controlled noise to the user-keyword matrix to satisfy differential privacy (47), the most popular privacy model lately. Finally, the perturbed user-keyword matrix is disclosed to the data consumer, where each user ID is anonymized. If the social graph corresponding to the data set is also needed, existing defenses such as (64; 94; 129; 103; 92) should be adopted to preserve both link privacy and vertex privacy. Our defense applies to a wide range of social media applications. For example, the data consumer can infer demographic information about the target population from the perturbed data set.

Our contributions can be summarized as follows.

- We are the first to coin the problem of privacy-preserving social media data publishing to the best of our knowledge, for which a system model is also proposed.
- We propose a novel mechanism to guarantee differential user privacy while maintaining high data utility in social media data publishing. The popular Laplacian mechanism to achieve differential privacy suffers from the *curse of dimensionality* (19) and can bring huge noise to the original dataset which significantly reduces data utility. We define a new metric called ϵ-text indistinguishability whereby to design a mechanism to break the constraint of the Laplacian mechanism.
- We thoroughly evaluate the proposed defense on a real-world dataset with regard to user privacy and data utility. Our results show that high-level privacy protection can be achieved without significantly sacrificing data utility. For example, we show that our mechnism can reduce the privacy leakage by as much as 64.1% by reducing only 1.61% of utility in terms of classification accuracy.

The rest of this chapter is organized as follows. Section 6.2 presents the problem statement. Section 6.3 illustrates our design for differentially privacy-preserving social media publishing. Section 6.4 evaluates the utility and privacy aspects of our mechanism by detailed experiments. Section 6.5 surveys the related work. Section 6.6 concludes the chapter.

6.2 Problem Statement

In this section, we first describe the system model for social media data publishing, followed by the adversary model. Next, we briefly argue that current publishing

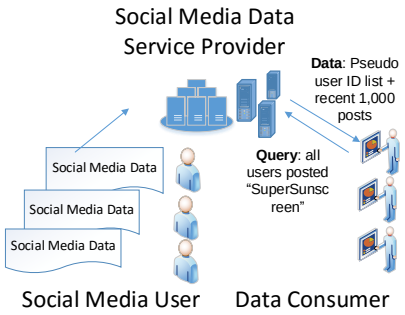


Figure 6.1: Social Media Data Publishing. The Data Consumer Submits a Query to Request the Data of Everyone Who Tweeted the Keyword "SuperSunscreen" in the Past 48 Hours. The Data Service Provider Then Return All the Qualifying Users with Anonymous IDs and Their Recent 1,000 Posts.

policies of major social media service providers are vulnerable under our adversary model. Finally, we state the design objectives.

6.2.1 Social Media Data Publishing

We consider a system with three parties: social media users, social media data service providers, and social media data consumers, as shown in Fig. 6.1.

- Social media users use the social media to connect with their friends and/or ones they have followed and generate the original texts which could be set either *private* or *public*. Public users are searchable either directly via the social media service provider's website or APIs or from the external tool such as Google. By setting his/her profile private, a private user only allows the authenticated users to access the profile and is not searchable from other users. However, the social media service provider still has full access to all the private and public data per user agreements.

- The social media data service provider (or DSP for short) hosts and provides most likely paid access to social media data. A DSP can be a social media service provider such as Twitter or Facebook itself. It can also be an emerging third-party data company such as Gnip or DataSift, which partners with social media service providers to provide social media data services. For example, Gnip and DataSift both have authorized access to Twitter's Firehost engine whereby to have access to complete, intact, and realtime Twitter data. The DSP can publish the data according to the privacy policies and agreements (detailed in Section 6.2.3) which users consent to when signing up for using social media services. Generally, the DSP has full rights to use all the hosted data for their businesses and also share the data with data consumers. For example, the DSP can sample the whole user space according to data consumers' requests, assign an anonymous ID to each sampled user, process the original data from each user according to data requests, and finally deliver the processed data to data consumers.

- Data consumers purchase social media data from the DSP whereby to run various social media mining algorithms to extract useful information. A data consumer can be an individual, a business, a government agency, a research institution, or any other entity in public and private sectors who is aware of the growing importance of social media data. A data consumer typically sends to the DSP a request specifying its query conditions, pays for the request, and then receives the data. For example, company A may request the data of all the users in the west coast who have tweeted the keyword "company A" in the past week. After receiving the data from the DSP, it can explore advanced social media mining algorithms to identify critical market trends and analyze the

users' demographic information such as age, location, education level, income level, marital status, occupation, religion, and family size.

We emphasize that a data consumer currently cannot obtain complete and intact social media data without the DSP's support. For example, social media service providers publish public APIs which allow a data consumer to crawl their platforms, but data crawling via public APIs has critical limitations that hinder the retrieval of complete data in need. First, each API call only returns a random and small sample of the whole data satisfying the query conditions. Second, such public APIs often have a rate limit on how many queries a data consumer can run in a particular time period. For example, Twitter users can make 180 requests/queries per 15 minutes to get others' timelines (136). Finally, a nontrivial set of users (e.g., 11.84% for Twitter (12) and 52.8% for Facebook (44)) set their data private such that unauthorized data consumers have no way to search for and obtain these private data. As a result, the data consumer needs to request the complete data in need from the DSP who has the complete dataset.

6.2.2 Adversary Model (User-Linkage Attack)

The DSP is assumed to be fully trusted by both social media users and data consumers. On the one hand, social media data are increasingly important profit venues of social media service providers. For example, it is reported that data licensing has already account for more than 10% of Twitter's total revenue. So the DSP is motivated to provide high-utility data services to data consumers. On the other hand, the DSP is obligated to protect the privacy of social media users due to legal concerns, public relations, and many other reasons. The framework we propose allow the DSP to strike a good balance between data utility and user privacy.

Some advanced social media users may be privacy-aware and have taken some

actions to protect their privacy. For example, the statistics in (12) and (44) show that 11.84% of Twitter users and 52.8% of Facebook users set their accounts private, respectively. As said, the DSP still has access to the complete data despite the users' privacy settings. In addition, the users' effort to protect their privacy fails in the presence of the attack outlined below.

Our focus is to defend against the *user-linkage* attack, which can be launched by a curious or even malicious data consumer. Assume that the DSP has anonymized every user ID in the dataset and also taken existing defenses such as (64; 94; 129; 103; 92) to guarantee link and vertex privacy. There are two possible versions of the user-linkage attack. In the first version, the attacker locates some target users by random browsing or searching via public APIs on the social media platform. It knows that these users must be in the received dataset under anonymous IDs. Existing defenses only consider link and vertex privacy via various obfuscation mechanisms, and no attention has been paid to text data. Armed with the text data of the target users with real IDs, the attacker can easily locate the corresponding anonymous IDs in the dataset. In the same way, the attacker can link the real IDs of the initial target users's friends to the corresponding anonymized IDs, and so on. The attacker eventually can uncover all the mappings between real and anonymous IDs in the dataset, despite the DSP's anonymization effort even based on existing advanced defenses (64; 94; 129). In the second version, the attacker tries to learn more information beyond the received dataset. It starts by finding some interesting posts/tweets in the anonymized dataset and then easily locating the real users by performing simple text matching on the social network platform. Once the real users are located, the attacker can learn their latest information and even other information the attacker's original query does cover.

To highlight the danger posed by such user-linkage attacks, assume that the attacker is an intelligent criminal. He first sends a query for all the users in a target

physical region who tweeted "vacation" in the past month. Then he locates valuable targets in the anonymized dataset by running sophisticated social media mining algorithms. Next, he uses the user-linkage attack to find out the real IDs of these valuable targets and track their realtime statuses on the social media platform. Finally, the attacker can well plan break-ins and thefts aiming the victims who mentioned future vacation plans. To the best of our knowledge, we are the first to notice and study the defense against such text-based user-linkage attacks which can be extremely dangerous.

6.2.3 Vulnerability of Current Social Media Data Publishing Policies

Social media data and their sharing policies are critical for the business of social media service providers. The typical business model is that users can use social media services for free, but in return they have to allow social media service providers to fully access, record, use, and profit from their public/private data. In this section, we review the current data publishing policies of Facebook and Twitter, two representative social media service providers and further highlight the need for privacy-preserving social media data publishing. Note that we have absolutely no intention to compare/critize/judge these data publishing policies which are in effect for various technical and non-technical reasons.

In particular, Facebook [1] claims that "We want our advertising to be as relevant and interesting as the other information you find on our Services. With this in mind, we use all of the information we have about you to show you relevant ads," and "We transfer information to vendors, service providers, and other partners who globally support our business, such as providing technical infrastructure services, analyzing how our Services are used, measuring the effectiveness of ads and services,

[1] `https://www.facebook.com/policy.php`

providing customer service, facilitating payments, or conducting academic research and surveys." Similarly, Twitter [2] allows third-party data consumers to access both the private and public user data, as they claimed "We engage service providers to perform functions and provide services to us in the United States, Ireland, and other countries. We may share your private personal information with such service providers subject to obligations consistent with this Privacy Policy and any other appropriate confidentiality and security measures, and on the condition that the third parties use your private personal data only on our behalf and pursuant to our instructions." These publishing policies are obviously subject to the aforementioned user-linkage attack, regardless of whether advanced defenses such as (103; 92) are incorporated to protect link and vertex privacy.

6.2.4 Design Objectives

We consider the following problem within the aforementioned social media data publishing framework. After receiving a data query from the data consumer, the DSP searches the entire social media database to generate a dataset $\mathcal{D}$, which contains all the users satisfying the query and their published texts (e.g., tweets, retweets, and replies) during the period specified in the query. Each user in $\mathcal{D}$ is assigned an anonymous ID to provide baseline user privacy. The data consumer may also request the social graph associated with $\mathcal{D}$, in which case we assume that existing defenses such as (64; 94; 129; 103; 92) are adopted to preserve link and vertex privacy such that it is infeasible to link an anonymous ID to the real user based on his/her vertex's graphical property in the social graph. Our focus is to let the DSP transform the raw dataset $\mathcal{D}$ into a new one $\mathcal{D}'$ by perturbing the user texts according to the following three requirements.

[2] `https://twitter.com/privacy?lang=en`

- *Completeness*: each data item in $\mathcal{D}$ can be mapped to a unique item in $\mathcal{D}'$, and vice versa. In other words, no user is added to or deleted from $\mathcal{D}$ to create $\mathcal{D}'$.

- *Privacy Preservation*: The user texts in $\mathcal{D}'$ can be used to link any anonymous ID in $\mathcal{D}'$ to the corresponding real user ID with negligible probability. This means that text-based user-linkage attacks can be thwarted with overwhelming probability.

- *High Utility*: $\mathcal{D}'$ and $\mathcal{D}$ should lead to comparable data utility at the data consumer when conducting common data mining tasks such as statistical aggregation, clustering, and classification.

6.3 Differentially Privacy-Preserving Social Media Data Publishing

In this section, we present a novel technique to achieve differentially privacy-preserving social media data publishing with the aforementioned design goals in mind. Inspired by geo-indistinguishability from (14), which is proposed to protect location privacy, we propose a novel notion of *text-indistinguishability* as the foundation of our technique. In what follows, we introduce the background on text modelling Section 6.3.1 and differential privacy in Section 6.3.2. We then present the concept of text-indistinguishability in Section 6.3.3, followed by our technique in Section 6.3.4. A working example is subsequently given in Fig. 6.3.5. Finally, we theoretically analyze our technique in Section 6.3.6 and make some remarks in Section 6.3.7.

6.3.1 Text Modeling

As stated before, social media service providers such as Facebook and Twitter currently publish the original data set $\mathcal{D}$ to the data consumer, which contains the complete and intact user texts. We assume that there are n users in $\mathcal{D}$, each assigned

an anonymous ID. There are two obvious drawbacks for this publishing method. First, although this method can enable the maximum data utility, it is vulnerable to the text-based user-linkage attack. Second, the data consumer cannot directly use the original texts which are highly unstructured and noisy, as mentioned in Section 6.1. For example, common machine learning algorithms such as SVM and K-means require the input for each user to be a vector. Therefore, from the perspectives of both privacy protection and data usability, the DSP needs to transform each user's texts into a numerical vector. Here we introduce text modeling, a standard process to achieve this goal.

We first remove *stop words* in a stop-word list, [3] in which the words such as "the" and "those" are considered more general and meaningless. Then we conduct stemming (116) to reduce inflected words to their stem forms such that the words with different forms can be related to the same word. For example, "play", "playing", and "played" are all reduced to "play".

Next, we represent the keyword space for the cleansed texts using a τ-gram technique, which is widely used for statistical text analysis. The τ-gram technique splits a give message into sequences of τ contiguous words, each referred to as a τ-gram with τ ranging from 1 to the message length. For example, consider a tweet {"#SuperSunscreen is really useful, and I like its smell"}. After removing stop words and performing stemming, we have {"supersunscreen really useful like smell"}. The corresponding 1-grams are {"supersunscreen", "really", "useful", "like", "smell"}, and the corresponding 2-grams are {"supersunscreen really", "really useful", " useful like", "like smell"}. We let $\mathcal{N}_i$ denote the τ-grams of $\mathcal{X}_i$ for each user $i \in [1, n]$ for all possible values of τ. Then we choose the top m most frequent τ-grams in $\bigcup_{1 \le i \le n} \mathcal{N}_i$, each of which is referred to as a *keyword* hereafter.

[3] `http://www.lextek.com/manuals/onix/`

Finally, we use Term Frequency Inverse Document Frequency (TF-IDF) (85) to derive each element $D_{i,j}$ in the eventual dataset. Specifically, let $\Gamma(j)$ be the number of times a τ-gram j appears in the τ-gram list $\mathcal{N}_i$ of user i, $\Gamma_i^* = \max_{j \in \mathcal{N}_i} \Gamma(j)$, and $\Gamma'(j)$ denote the number of users whose τ-gram lists contain j. We define

$$D_{i,j} = (0.5 + 0.5 * \frac{\Gamma(j)}{\Gamma_i^*}) * \log(\frac{n}{\Gamma'(j)}) \ . \tag{6.1}$$

The above normalization is necessary because the users normally have very different tweet sets and thus different τ-gram lists. Interested readers are referred to (85) for more details about TF-IDF. We abuse the notation by letting $\mathcal{D} = [D_{i,j}] \in \mathbb{R}^{n \times m}$ denote the dataset after text modeling as well, which is essentially an $n \times m$ user-keyword matrix. We also let $U_i := \langle D_{i,1}, \ldots, D_{i,m} \rangle$ denote the text vector of user i ($i \in [1, n]$), i.e., the ith row in $\mathcal{D}$.

It is a common practice to use 1-grams and 2-grams only for high computational efficiency without significantly sacrificing the analysis accuracy. So the keyword space and user-keyword matrix can be constructed very quickly in practice. Also note that the DSP needs to publish the $\tau-$gram name of each column. Otherwise, the data consumer has no idea about the physical meaning of the released data.

6.3.2 Why Differential Privacy?

The text model above has two important implications. First, it makes the unstructured social media data structured by reducing the keyword dimension from unlimited to m. Second, since the keyword space is composed of the top m most frequent τ-grams, the users' privacy has been largely improved in contrast to the original intact text data. For example, when a user has a tweet saying "The last class with students at CSE561, #MIT", the word "CSE561" or even "MIT" has very low probability to be selected in the keyword space. Therefore, this critical information

has been hidden by the text modeling process. The privacy threat, however, cannot be completely eliminated. For instance, the 1-grams such as "last", "class", and "student" may still be released. These pieces of information can at least tell that the user is a professor or teacher. By combining other text information such as "computer" and "software," the attacker can further link the target user to a college professor teaching computer science. Such inferences can be continued until the target is linked to one or a few real IDs on the social media platform.

Differential privacy is a powerful technique to protect such linkage attacks. Proposed by Dwork *et al.*(47; 48), differential privacy protects the individual user's privacy during the statistical query over a database. If each user in the database is independent, [4] with any side information except the target him/herself, the attacker cannot infer whether the target user is in the database or which record is associated with him/her (74). Providing arguably the strongest analytical protection for user privacy, the differential privacy model can be more formally defined as follows, which is tailored for our social media data publishing framework.

Definition 6.3.1 (ϵ-Differential Privacy (47))**.** *Given a query function $f(\mathcal{D})$ with an input dataset $\mathcal{D} \in \mathbb{R}^{n \times m}$ and a desirable output range, a mechanism $K(\cdot)$ with an output range $\mathcal{R}$ satisfies* ϵ-differential privacy *iff*

$$\frac{Pr[K(f(D_1)) = R \in \mathcal{R}]}{Pr[K(f(D_2)) = R \in \mathcal{R}]} \leq e^{\epsilon} \tag{6.2}$$

for any datasets $D_1, D_2 \in \mathbb{R}^{n \times m}$ that differ on only one row.

Here ϵ is called the privacy budget. Large ϵ (e.g. 10) results in large e^{ϵ} and indicates that the DSP can tolerate large output difference and hence large privacy loss (because the adversary can infer the change of the database according to the large

[4]The dependence among the users will hurt the privacy guarantee as indicated in (74; 91).

change of the query function $f(\cdot)$. By comparison, small ϵ (e.g., 0.1, $e^{0.1} = 1.1052$) indicates that the DSP can tolerate small privacy loss.

Differential privacy models can be interactive and non-interactive. Assume that the data consumer intends to execute a number of statistical queries on the same dataset. In the interactive model, the data consumer submits to the DSP the conditions for constructing the dataset $\mathcal{D}$ and also a desirable statistical query function f. Instead of returning $\mathcal{D}$ to the user, the DSP only responds with $K(f(\mathcal{D}))$, where $K(\cdot)$ perturbs the query result. In contrast, the DSP in the non-interactive model designs a mechanism $K(\cdot)$ to transform the original dataset $\mathcal{D}$ into a new dataset $\mathcal{D}' = K(f(\mathcal{D}))$. Finally, $\mathcal{D}'$ is returned to the data consumer which can execute arbitrary statistical queries locally.

6.3.3 ϵ-Text Indistinguishability: a New Notion

Our problem can be formulated according to a non-interactive differential privacy model as follows. Let us use an identity query $f_I(\cdot)$ as the query function such that $f(\mathcal{D}) = \mathcal{D}$. Our goal is to find a mechanism $K(\cdot)$ to transform the original user-keyword matrix (or dataset) $\mathcal{D}$ into a new one $\mathcal{D}' = K(\mathcal{D})$ such that ϵ-differential privacy can be achieved. Instead of transforming the entire dataset $\mathcal{D}$ as a whole, a more straightforward approach is to perform the transformation for each row individually, i.e., adding noise to each row $U_i \in \mathcal{D}$ to produce a new row $U_i' \in \mathcal{D}'$.

The Curse of Dimensionality. The Laplacian mechanism (47) is a popular technique for providing ϵ-differential privacy, but it suffers from the *curse of dimensionality*. To see it more clearly, recall that ϵ-differential privacy is defined over the query function f and unrelated to the dataset because Eq. (6.2) holds for all possible datasets. What matters is the maximum difference between $f(D_1)$ and $f(D_2)$ ($\forall D_1, D_2 \in \mathbb{R}^{n \times m}$), which is called the *sensitivity* of the query function f defined as

follows,

$$S(f) = \max \|f(D_1) - f(D_2)\|_1 \ . \tag{6.3}$$

For the identity query $f_I(\cdot)$ which transforms each text vector (row) in $\mathcal{D}$ to a new vector in $\mathcal{D}'$, the sensitivity can be furthered defined as

$$S(f_I) = \max \|U_i - U_j\|_1 \tag{6.4}$$

where $U_i \in \mathbb{R}^m$ and $U_j \in \mathbb{R}^m$ are any two arbitrary vectors based on TF-IDF (see Eq. 6.1).

The Laplacian mechanism can achieve ϵ-differential privacy by adding the Laplacian noise to the query result (47), i.e.,

$$K_{Lp}(f_I(U_i)) = U_i + (Y_{i1}, \cdots, Y_{im}), i = 1, \ldots, n \ , \tag{6.5}$$

where Y_{ij} are drawn i.i.d. from $\text{Lap}(S(f_I)/\epsilon) \propto e^{-\epsilon|x|/S(f_I)}$.

The Laplacian mechanism unfortunately decreases the utility of the transformed dataset. Specifically, the larger the dimension m from the output of the identity query function $f_I(\cdot)$, the larger the sensitivity $S(f_I)$, the larger deviation of the Laplacian noise. Moreover, the large noise accumulated from the high dimension will be added to each single element of $K_{Lp}(f_I(U))$, leading to the so-called curse of dimensionality. Specifically, from the definition of the text vector U_i in Eq. (6.1), the norm of each element in U_i should be less than $\log(n)(\approx 11.5$ when $n = 100000)$. When the dimension m (e.g., 10000) is large enough, the added Laplacian noise has deviation $O(m)$, which can easily exceed the norm of original text element (≈ 11.5).

ϵ-Text Indistinguishability. The root cause of the curse of dimensionality is that the noise added to a single element in every text vector U_i ($\forall i \in [1, n]$) is proportional with the L_1-sensitivity of U_i. To tackle this problem, we need to limit the sensitivity of the whole text vector to the norm of the vector, instead of the individual element.

To begin with, we need to generalize the concept of differential privacy defined in Definition 6.3.1. The generalization of differential privacy was first proposed by Andrés *et al.* for location privacy (14), where the privacy budget is proportional to the physical distance between any two users. They also propose the concept of geo-indistinguishability such that the service provider reports similar distribution with the difference bounded by $e^{\epsilon d(\text{loc}_1, \text{loc}_2)}$ for any two users at locations loc_1 and loc_2, respectively. Inspired by this work, we let $d(U_i, U_j)$ denote the Euclidean distance between U_i and U_j, which are any pair of text vectors in the user-keyword matrix $\mathcal{D}$. We further redefine the privacy budget as $\epsilon d(U_i, U_j)$ and propose the notion of ϵ-text indistinguishability as follows.

Definition 6.3.2 (ϵ-Text Indistinguishability)**.** *Given the user-keyword matrix* $\mathcal{D} = [D_{i,j}] \in \mathbb{R}^{n \times m}$*, a mechanism* $K_t(\cdot)$ *satisfies* ϵ-Text Indistinguishability *iff*

$$\frac{Pr[K_t(U_i) = U^* \in \mathbb{R}^m | U_i]}{Pr[K_t(U_j) = U^* \in \mathbb{R}^m | U_j]} \leq e^{\epsilon d(U_i, U_j)} \ , \tag{6.6}$$

where U_i *and* U_j *are any user text vector pair in* $\mathcal{D}$*, and* U^* *is a text vector in perturbed user-keyword matrix* $\mathcal{D}'$*.*

The above definition means that any two vectors U_i and U_j in $\mathcal{D}$ can be transformed (or perturbed) by the mechanism $K_t(\cdot)$ into the same vector in $\mathcal{D}'$ with probability $\geq 1 - e^{\epsilon d(U_i, U_j)}$. In other words, the more similar two text vectors are, the more non-distinguishable they are after transformation, and vice versa. The maximum privacy budget is given by $\epsilon r_{\max}$, where $r_{\max}$ denotes the maximum Euclidean distance between two text vectors in $\mathcal{D}$. As in the original ϵ-differential privacy mechanism, the larger the privacy budget, the larger the privacy loss the DSP can tolerate, and vice versa. The following theorem gives the upper bound of $\epsilon r_{\max}$, based on which the DSP can select ϵ to ensure an acceptable privacy budget.

Theorem 6.3.1. *Given the user-keyword matrix $\mathcal{D} \in \mathbb{R}^{n \times m}$ built according to Eq. (6.1), the maximum Euclidean distance between two text vectors is $r_{\max} \leq \sqrt{m}\log(n)$.*

Proof. According to the definition in Eq. (6.1), a text vector U has the maximum norm $\sqrt{m}\log(n)$ when each of its element is equal to the maximum value $\log(n)$. It follows that $r_{\max} \leq \|U_1 - U_2\| \leq \|U\| \leq \sqrt{m}\log(n)$. □

The upper bound above is almost unreachable in practice, as it requires that all the m keywords be used by only one user. So $r_{\max}$ in practical scenarios is far less than $\sqrt{m}\log(n)$. But if the DSP chooses ϵ according to $\sqrt{m}\log(n)$, the effective privacy budget for many text-vector pairs is actually very small, implying that these text-vector pairs are very likely to be indistinguishable after perturbation.

6.3.4 Achieving ϵ-Text Indistinguishability

In this section, we propose a mechanism to achieve the ϵ-text indistinguishability. To this end, we first assume $r_{\max}$ to be infinite and then finite.

Mechanism for Infinite $r_{\max}$

The mechanism $K_t(f_I(\cdot))$, designed for the identity query $f_I(\cdot)$, maps each text vector $U \in \mathbb{R}^m$ of the dataset $\mathcal{D}$ to a new U' with the same dimension m. To that end, we write the perturbed U' as:

$$U' = U + d\Theta$$

where d is a random variable indicating the Euclidean distance between U and U', and Θ is an m-dimensional random vector drawn from the m-dimensional unit hypersphere. The mechanism is then composed of two steps: the generation of the magnitude and the direction. Since the drawing of Θ is straightforward, we focus on generating d. Similar to the Laplacian mechanism (47), we let d deviate from the

center U according to the Laplacian distribution,

$$g(d) = \epsilon e^{-\epsilon d} \tag{6.7}$$

where d ranges from zero to infinity. It is easy to check that $\int_0^{+\infty} g(d) = 1$.

The CDF of d is given by

$$C_\epsilon(d = r) = \int_0^r \epsilon e^{-\epsilon x} dx = 1 - e^{-\epsilon r}. \tag{6.8}$$

The CDF above tells us how to generate a random d. Specifically, given a user text vector U, we want to generate a perturbed vector which has at most d Euclidean distance from U. Since d follows the PDF defined in Eq. (6.8), given a random probability $p \in [0, 1]$, we can obtain

$$d = C_\epsilon^{-1}(p) = -\frac{\log(1-p)}{\epsilon}\ . \tag{6.9}$$

We now show that the proposed mechanism satisfies ϵ-text indistinguishability.

Theorem 6.3.2. *The mechanism $K_t(f_I(\cdot))$ defined above achieves the ϵ-text indistinguishability.*

Proof. Given two user text vectors U_i and U_j, the probability quotient of being perturbed the same vector U^* can be computed as

$$\begin{aligned} \frac{Pr[U = U^*|U_i]}{Pr[U = U^*|U_j]} &= \frac{Pr[d(U^*, U_i)]\Theta_1}{Pr[d(U^*, U_j)]\Theta_2} \\ &= e^{\epsilon(d(U^*,U_i) - d(U^*,U_j))} \leq e^{\epsilon(d(U_i,U_j))}. \end{aligned} \tag{6.10}$$

Here Θ_1 and Θ_2 can be canceled because both are drawn from the m-dimensional unit hypersphere with the same probability, and the inequity holds because of the triangle inequity. Therefore, the conclusion holds. □

Mechanism for Limited $r_{\max}$

The mechanism $K_t(f_I(\cdot))$ in last section maps the user text vector U to U' with potentially infinite distance. However, we have demonstrated in Theorem 6.3.3 that any text vector pair have the Euclidean distance bounded by $r_{\max}$. Here we present how to truncate the mapping into a specific $r_{\max}$. We denote the corresponding mechanism as $K_r(f_I(\cdot))$.

As we can see from Fig. 6.2, $C_\epsilon(d = r)$ will approach to one quickly as r increases.

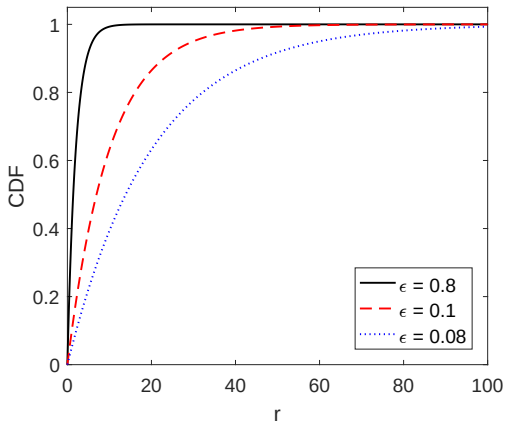


Figure 6.2: The CDF of d with Different ϵs.

Therefore, we define a tolerance parameter γ to indicate how much of CDF will be outside $r_{\max}$. In other words,

$$1 - \gamma = C_\epsilon(d = r_{\max}) = 1 - e^{-\epsilon r_{\max}}$$

$$\Rightarrow \epsilon = -\frac{\log(\gamma)}{r_{\max}}. \tag{6.11}$$

The algorithm of $K_r(f_I(\cdot))$ is listed in Alg. 5. Given the tolerance parameter γ and $r_{\max}$, we compute the ϵ according to Eq. 6.11 in Line 1. Then for each U_i in the dataset $\mathcal{D}$, we draw the noise vector $d\Theta$ by two steps: (1) obtain the magnitude randomly in Line 3 by using $r = C_\epsilon^{-1}(p)$ where p is the random CDF, and (2) compute

the direction Θ in Line 4 by drawing a random vector from the unit m-dimensional hypersphere. Line 5 adds the noise to U to get U'.

Algorithm 5: Perturbation algorithm for mechanism $K_r(f_I(\cdot))$

input : $r_{\max}, \gamma, \mathcal{D} = \{U_1, \ldots, U_n\}$

output: Perturbed dataset $\mathcal{D}' = \{U'_1, \ldots, U'_n\}$

1 Compute ϵ according to Eq. (6.11);

2 For each $U_i, i = 1, \ldots, n$;

3 Select a random number $p \in [0, 1]$, and compute the radius d according to Eq. (6.9) ;

4 Select a random vector $N \in \mathbb{R}^m$, and normalized it to have unit L_2 norm, i.e., $\Theta = N/\|N\|_2$;

5 $U'_i = U_i + d\Theta$;

We also show that the mechanism $K_r(f_I(\cdot))$ achieves the ϵ-text indistinguishability.

Theorem 6.3.3. *The mechanism $K_r(f_I(\cdot))$ defined above achieves the ϵ-text indistinguishability within $r_{\max}$.*

Proof. For each text vector U, with the probability of $1-\gamma$, the perturbed text vector U' has the Euclidean distance less or equal to $r_{\max}$ from U. The rest steps follow the proof of Theorem 6.3.2, and the conclusion holds. □

6.3.5 A Working Example

Fig. 6.3 gives a simple example for our system operations. Here the data consumer wants to find the users who have posted about a sunscreen product "SuperSunscreen". The DSP finds three qualified users with tweets as "Use #SuperSunscreen with mom,

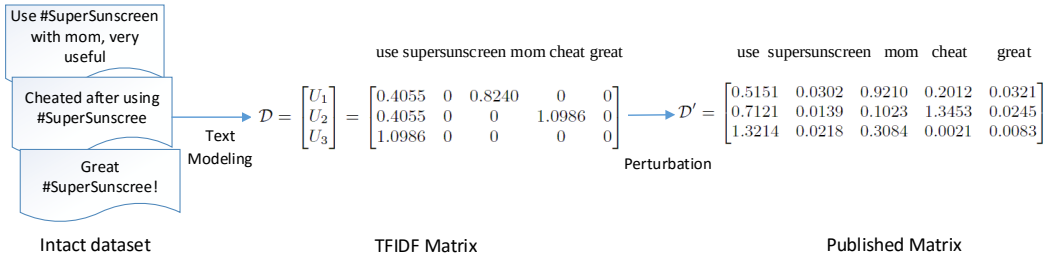


Figure 6.3: The Illustration of Differentially Privacy-preserving Social Media Data Publishing. Given Three Users with Intact Dataset, We First Use the Text Model to Build a Matrix $\mathcal{D}$, Then Add the Controlled Noise, and Finally Release the Perturbed Matrix $\mathcal{D}'$ and the Keywords in Each Column.

very useful", "Cheated after using #SuperSunscree ", and "Great #SuperSunscree!", respectively.

The first step is to remove the stop words, conduct the stemming, and then compute the TF-IDF matrix defined by Eq. 6.1. We obtain the 1-gram lists for these three users as "use(2), SuperSunscreen, mom", "cheat, use, SuperSunscreen", and "great SuperSunscreen", respectively. Consider the 1-grams keyword space as "use, SuperSunscreen, mom, cheat, great", we can build the TF-IDF matrix $\mathcal{D}$ as follows.

$$\mathcal{D} = \begin{bmatrix} U_1 \\ U_2 \\ U_3 \end{bmatrix} = \begin{bmatrix} 0.4055 & 0 & 0.8240 & 0 & 0 \\ 0.4055 & 0 & 0 & 1.0986 & 0 \\ 1.0986 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Then we apply the perturbation algorithm in Alg. 5 to generate a perturbed matrix $\mathcal{D}'$ as follows:

$$\mathcal{D}' = \begin{bmatrix} 0.5151 & 0.0302 & 0.9210 & 0.2012 & 0.0321 \\ 0.7121 & 0.0139 & 0.1023 & 1.3453 & 0.0245 \\ 1.3214 & 0.0218 & 0.3084 & 0.0021 & 0.0083 \end{bmatrix}.$$

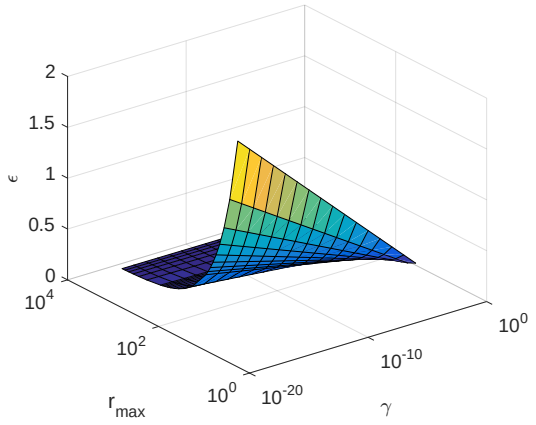


Figure 6.4: Determine ϵ by γ and $r_{\max}$.

The DSP then release $\mathcal{D}'$ and the keyword space to the data consumer for machine learning tasks. As we can see, the perturbed matrix largely reduce the distance between each user text vector by sacrificing little utility. We will evaluate the tradeoff between the privacy and utility later.

6.3.6 Performance Analysis

Privacy budget

As shown in Eq. (6.11), ϵ, which is the constant scale of the privacy budget ϵd, can be determined by γ and $r_{\max}$, as shown in Fig. 6.4. As we can see, for $r_{\max}$ from 1 to 10^4 and γ from 1 to 10^{-16}, ϵ is always less than 2. Moreover, since ϵ is reversely proportional to $r_{\max}$, the whole privacy budget ϵd is less than $-\log(\gamma)$ (because $d \leq r_{\max}$), which is relatively small. The small privacy budget is critical for differential privacy mechanisms because a large budget result in a large privacy loss.

Break the Curse of Dimensionality

As stated in Section 6.3.3, the original ϵ-differential privacy notion and the corresponding Laplacian mechanism suffer the curse of dimensionality for perturbing the text vector. The reason is that the noise strength added to each element in the text vector has a scale of $S(f_I)/\epsilon$, where $S(f_I)$ is the L_1 sensitivity of the text vector and proportional to the dimension m. We now estimate the scale of the noise strength for the mechanism $K_r(f_I(\cdot))$.

Theorem 6.3.4. *Given a text vector $U \in \mathbb{R}^m$, by applying the mechanism $K_r(f_I(\cdot))$, the expected noise strength for each element in U is unrelated to the dimension m.*

Proof. According to Eq. (6.9), we can obtain

$$E(d) = \int_0^1 -\frac{1-p}{\epsilon} dp = \frac{1}{2\epsilon}.$$

Combining Eq. (6.11) and the upper bound of $r_{\max}$ in Theorem 6.3.3, we can obtain

$$E(d) = \frac{2r_{\max}}{\log(\gamma)} < -\frac{2\sqrt{m}\log(n)}{\log(\gamma)}.$$

Here $E(d)$ is the expected noise strength for the whole vector U. Therefore, the expected strength for each element in U is

$$\frac{E(d)}{\sqrt{m}} < -\frac{2\log(n)}{\log(\gamma)}$$

which is unrelated to the dimension m. □

Moreover, by setting the γ to extremely small (e.g., 10^{-16}), the upper bound approaches 1. Note that according to Theorem 6.3.3, this upper bound is rather loose. Therefore, the expected noise strength for each element is far less.

(α, δ)-usefulness

The mechanism $K_r(f_I(\cdot))$ also satisfies (α, δ)-usefulness defined by (25).

Definition 6.3.3 ((α, δ)-usefulness)**.** *A ϵ-text indistinguishability mechanism K satisfies (α, δ)-usefulness iff for every user text vector U, with the probability at least δ, the perturbed text vector U' satisfies $d \leq \alpha$.*

Theorem 6.3.5. *The mechanism $K_r(f_I(\cdot))$ defined above achieves the (α, δ)-usefulness.*

Proof. It can be easily seen according to the CDF of the random variable d in Eq. (6.8). □

6.3.7 Remarks

We have three remarks to make on our design.

First, the proposed ϵ-text indistinguishability and $K_r(f_I(\cdot))$ mechanism can be used to protect the privacy in other contexts as long as the dataset $\mathcal{D}$ is homogeneous, meaning that each column should have a similar numerical range. The reason is that the model generalizes and represents the whole vector by its L_2 norm, while ignoring the difference for each element in the vector. If the element in each row vector is different, the generalization could become meaningless. Nevertheless, the proposed mechanism break the curse of dimensionality, and it is an interesting topic to extend the proposed model to heterogenous datasets.

Second, the mechanism in Alg. 5 could generate a perturbed vector with very large norm ($> r_{\max}$). However, the attacker cannot obtain any external information from this abnormality. The reason is that the probability for any user who has this abnormality is random. Moreover, the probability is negligible. For example, when

$r_{\max} = 100$ nd $\gamma = 10^{-8}$, the probability of abnormality is 10^{-8}. In general, the less γ, the less the abnormality, and vice versa.

Third, $r_{\max}$ is important for our mechanism, and we can further reduce $r_{\max}$ by clustering to define the (k, ϵ)-text indistinguishability. Specifically, we can partition the dataset into k clusters by grouping the close vectors in a cluster, and then conduct the ϵ-text indistinguishability independently within each cluster. Since each local group has a smaller $r_{\max}$, less noise is added to the original dataset, leading to higher data usability. Obviously, the larger the K, the less the privacy and the higher the usefulness, and vise versa. We leave this extension as future work.

6.4 Evaluation

In this section, we use both a real-world dataset and simulations to evaluate the proposed ϵ-text indistinguishability mechanism $K_r(f_I(\cdot))$ in three aspects: the privacy and usefulness, the utility on a typical machine learning task, and the defense against the user-linkage attacks.

6.4.1 Dataset

As stated before, the data consumer aims to use the social media data to do the demographics analysis. Here we use a ground truth Twitter user dataset with known age information similar to (158). Specifically, a user A has age x if one of his friends has posted a tweet with the format "Happy x-birthday to A". We used Twitter Streaming API to monitor these tweets and the ground-truth users. We then manually check the consistency between the claimed age information and the tweets they have posted to. Finally, we found 5,710 users, which consist of 2,855 users who are at least and less than 25 years old, respectively. We crawled one year of their recent tweets and obtained 3,363,706 tweets. We then removed the stopping words

and conducted the stemming as stated in Section 6.3.1, and built the TF-IDF matrix according to Eq. (6.1) for the following experiments. Because of the randomness during the noise generation, we run each of the experiment 100 times and report the average results.

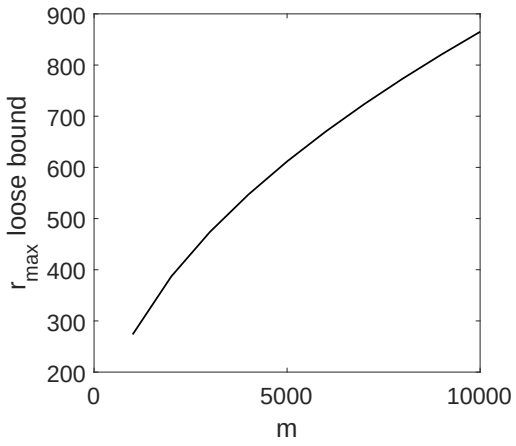


Figure 6.5: The Loose Upper Bound of $r_{\max}$.

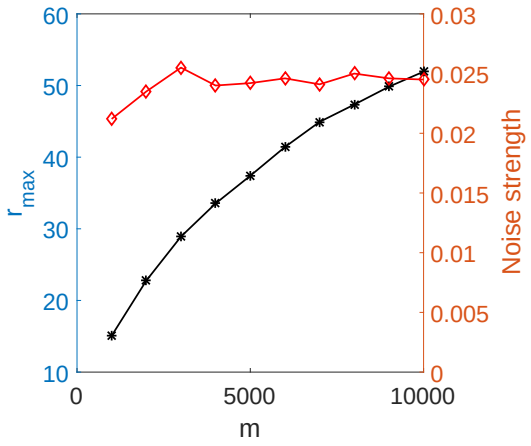
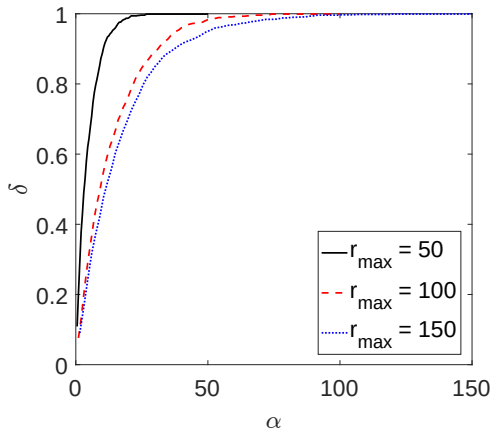
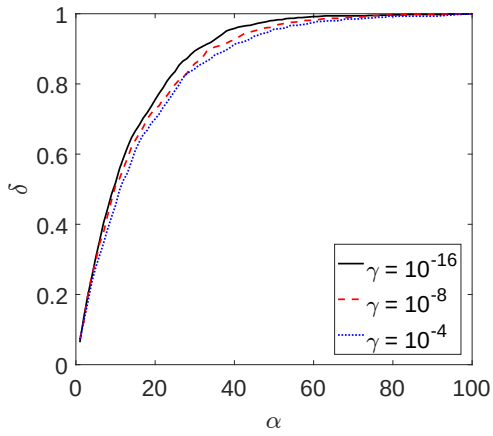


Figure 6.6: The Real $r_{\max}$ and the Noise Strength for Each Element.



(a) $\gamma = 10^{-8}$



(b) $r_{\max} = 100$

Figure 6.7: The Usefulness of the Mechanism.

6.4.2 Privacy and Usefulness

We first check $r_{\max}$ in the real-world dataset above. To start with, Fig. 6.5 shows the loose upper bound with different dimension m stated in Theorem 6.3.3. We set the number of users $n = 5710$. The upper bound is a sublinear function with m, and increases from 200 to 1000 when m ranges from 1000 to 10000.

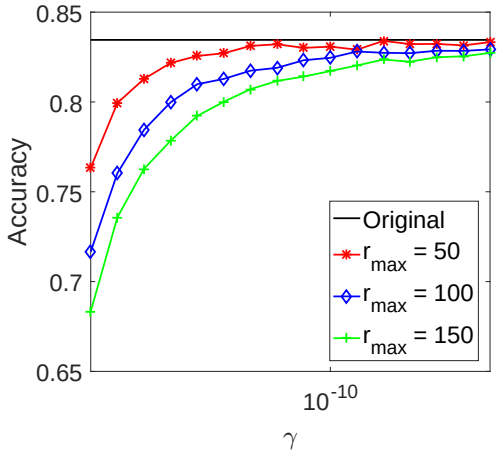
We also measure $r_{\max}$ in the dataset as shown in Fig. 6.6. Specifically, we compute $r_{\max}$ as the maximum L_2 norm of each row vector from the dataset D. As we can see, although the $r_{\max}$ increases sublinearly with m, the magnitude is much less than the upper bound in Fig. 6.5. The reason is twofold. First, because we built the TF-IDF dataset by choosing the most m frequent grams, the IDF term in Eq.(6.1) is much less than $\log(n)$. Second, the TF part is less than $\sqrt{m}$ because the text vector is sparse (each user has only used limited grams when m is large).

Given $r_{\max}$, Fig. 6.6 demonstrates the expected noise strength added for each single element in the text vector. As we can see, the noise strength is fairly stable with the dimension m, which is consistent with Theorem 6.3.4. Moreover, the expected noise strength ranges from 0.02 to 0.03, and is comparable to the original data. Therefore, the proposed mechanism can tolerate an arbitrary dimension, i.e., breaking the curse of dimensionality.

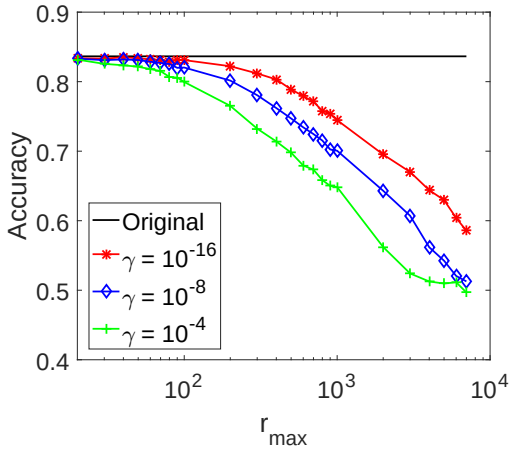
Fig. 6.7a and Fig. 6.7b show the (α, δ)usefulness of the mechanism at different $r_{\max}$ and γ, respectively. As we can see, with probability δ, the distance between the original text vector and the perturbed vector is within α, which verifies Theorem 6.3.5.

6.4.3 Performance on Classification

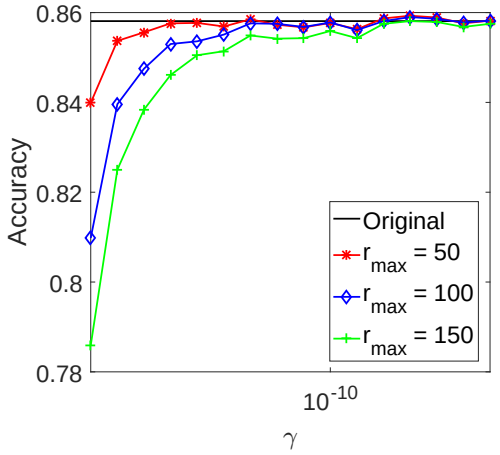
We evaluate the mechanism on classification, one of the typical applications from the machine learning community. As stated before, each user has the ground-truth



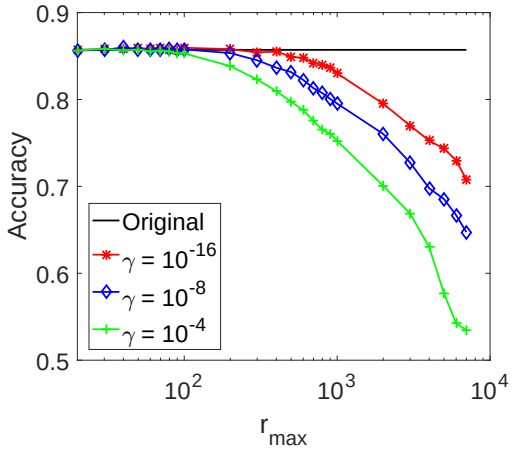
(a) $m = 1000$



(b) $m = 1000$



(c) $m = 5000$



(d) $m = 5000$

Figure 6.8: The Performance of Classification.

age information. We can then build a binary classifier to determine whether a user is younger than 25 years old or not. We use the Support Vector Machine (SVM) (128) algorithm to evaluate the classification performance on both the original and the perturbed datasets by ten-fold cross validation.

Fig. 6.8a demonstrate the accuracy with a changing γ. The straight and crooked lines represent the original and perturbed datasets, respectively. As we can see, the

smaller γ, the higher the performance for the perturbation mechanism. This result is not surprising. Specifically, Theorem 6.3.4 indicates that the smaller γ, the less the noise added to the original dataset. However, small γ will increase the privacy budget scale ϵ and hence the privacy loss.

Fig. 6.8b demonstrates the accuracy of the original dataset (straight line) and the perturbed datasets with different $r_{\max}$ (crooked curves). As we can see, the smaller $r_{\max}$, the better the accuracy because smaller $r_{\max}$ will incur less noise. However, less noise will cause a high privacy loss because the attacker can infer the victim given the huge difference between two perturbed vectors.

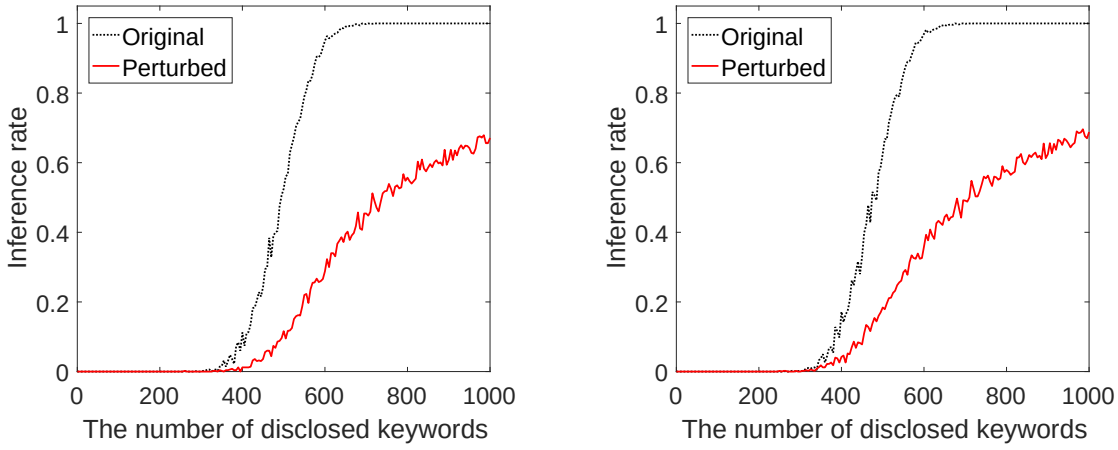
Fig. 6.8c and Fig. 6.8d show the classification performance on $m = 5000$. As we can see, both figures show the similar trend for $m = 1000$, meaning that the mechanism works well at various dimensions. Moreover, the performance when $m = 5000$ is slightly better than that when $m = 1000$. The reason is that more keywords lead to better classification.

6.4.4 Defense Against User-Linkage Attacks

Our mechanism is designed to defend against the user-linkage attack. The definition of ϵ-text indistinguishability in Definition 6.3.2 and the corresponding mechanism in Alg. 5 show that any user can be perturbed to other text vector with certain probability. Therefore, the perturbation can make the user-linkage attack more difficult to conduct. To evaluate the effectiveness of our mechanism, we need to model the strength of the attacker in terms of user inference. We consider two attack models here.

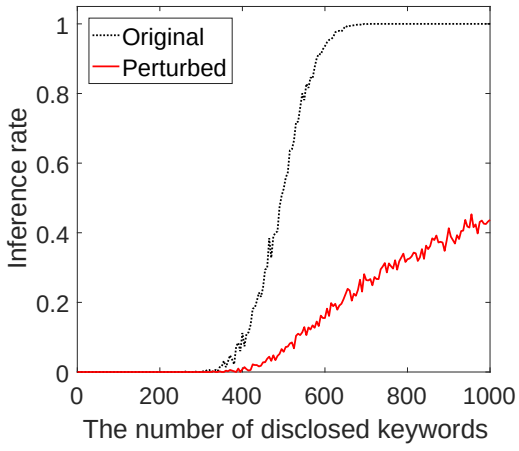
In *inference attack I*, we assume that the attack knows t elements of the victim's text vector, and t vary from 0 to m. We then build an estimated vector U' by keeping these t elements and setting other unknown elements to zero, and check whether the

estimated vector U' is in the K-nearest vector set in both the original D and the perturbed D'. It is expected that the larger the t, the stronger the attack, the higher the inference rate. We set $m = 1000$ and $\gamma = 10^{-8}$. We conduct the experiment by 1000 times and report the average results.

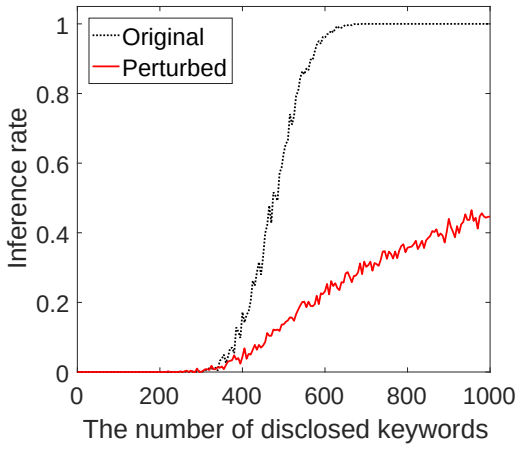


(a) $K = 1, r_{\max} = 100$

(b) $K = 10, r_{\max} = 100$



(c) $K = 1, r_{\max} = 200$



(d) $K = 10, r_{\max} = 200$

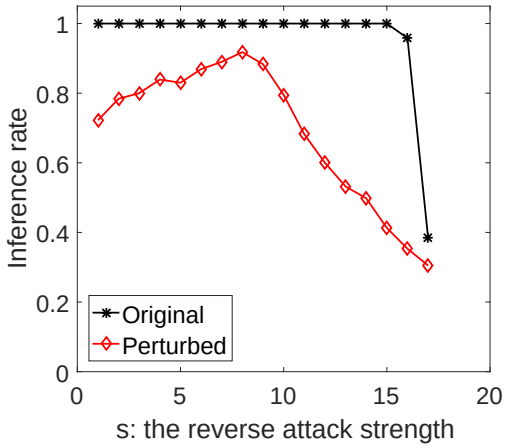
Figure 6.9: The Performance of Inference Attack I.

Fig. 6.9 shows the inference rate among the 1-nearest and 10-nearest vectors for $r_{\max} = 100$ and 200. We can make two observations. First, all the four curves show

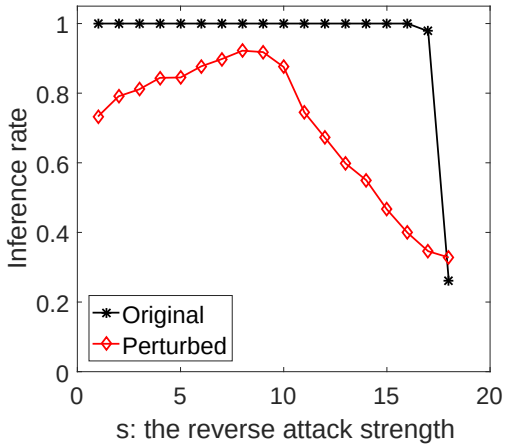
that the perturbation algorithm makes the user linkage attack much more difficult. Specifically, when the t increase from 300 to 600, the inference rate increase quickly from 0 to 100% for the original dataset. The inference rate then stay at approximate 100% when t is larger than 600. By comparison, the inference rate for the perturbed dataset is at most 68.8% for $r_{\max} = 100$ ($K = 10$) and 44.6% for $r_{\max} = 200$ ($K = 10$), respectively. Second, Fig. 6.9 demonstrate the tradeoff between the privacy and usefulness in terms of $r_{\max}$. Specifically, on the one hand, the mechanism's inference rate for $r_{\max} = 200$ is less than the rate for $r_{\max} = 100$. The reason is that larger $r_{\max}$ results in larger noise and hence higher-level privacy protection. On the other hand, larger $r_{\max}$ results in lower classification performance as indicated in Fig. 6.8. The tradeoff also holds in terms of γ.

In *inference attack II*, we assume that the attack knows the noisy but the whole text vector of the victim. To that end, we randomly select a victim vector U^* from D, add a noise vector N with the magnitude s where $1/s$ is the attack strength, and then check whether the noisy vector $\tilde{U} = U^* + N$ is in the K-nearest vector set in both the original D and the perturbed D'. We use the Euclidean distance to represent the difference between any vector pair. Obviously, it is expected that the weaker the attack strength, the higher the inference rate.

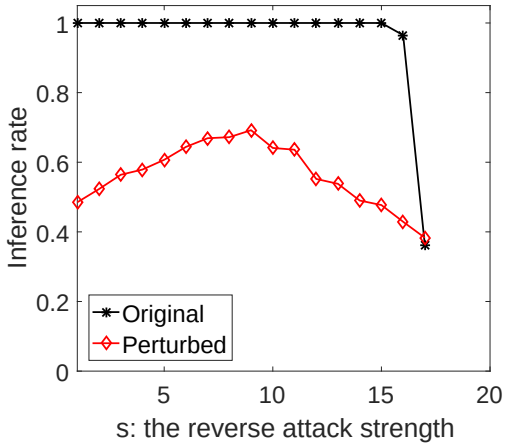
Fig. 6.10 show the inference rate among the 1-nearest and 10-nearest vectors for $r_{\max} = 100$ and 200. We can make the similar observations as in the inference attack I. First, the perturbation algorithm makes the user linkage attack much more difficult. Specifically, when the reverse attack strength s increases, the inference rate for the perturbed dataset decreases to about 30% for $K = 1$ and 40% for $K = 10$, meaning that the attacker has limited power to infer the victim. By comparison, the inference rate for the original dataset is always 100% when s is less than 17. The reason is each user text vector is very distinguishable. When $s > 17$, the inference for the original



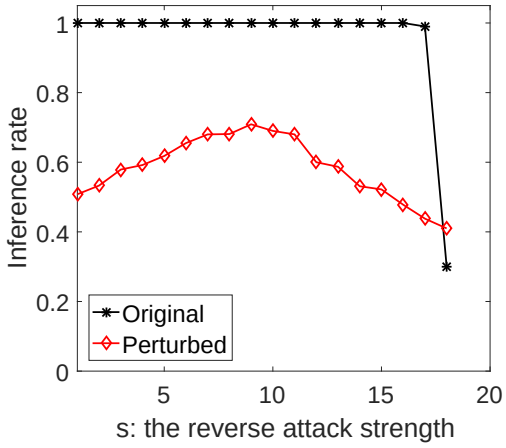
(a) $K = 1, r_{\max} = 100$



(b) $K = 10, r_{\max} = 100$



(c) $K = 1, r_{\max} = 200$



(d) $K = 10, r_{\max} = 200$

Figure 6.10: The Performance of Inference Attack II.

dataset decreases dramatically because the measured $r_{\max}$ for this dataset is 15.1 for $m = 1000$, as indicated in Fig. 6.6. Second, Fig. 6.9 demonstrate the tradeoff between the privacy and usefulness in terms of $r_{\max}$.

Moreover, users' privacy has not largely sacrifice the utility. For example, as a typical setting, when $r_{\max} = 100$ and $\gamma = 10^{-8}$, the inference rate with $s = 15$ and $K = 10$ is 47.7%, and the classification accuracy is reduced by only 1.61%. Therefore,

the proposed mechanism can achieve high privacy guarantee with little utility loss.

Note that there is a peak point for the inference rate on the perturbed dataset in Fig. 6.10. This is because that the perturbation also adds the noise vector in the similar way as in the inference attack II. For different $r_{\max}$, the perturbed vectors have different Euclidean distance from the original vectors. Recall that U' and $\tilde{U}$ are the perturbed vector and the estimated vector from the attacker for the victim U^*, respectively. When the difference between $d(U^*, \tilde{U})$ and $d(U^*, U')$ is small, the inference rate will increase. However, in reality, the attacker has little knowledge on the whole text vector for the victim. Therefore, it is difficult for them to conduct this type of inference.

6.5 Related Work

This section reviews the most relevant work.

6.5.1 Privacy on Social Media Platforms

Social media platforms host the network and also text information. The privacy threat of both pieces of information has been studied widely. For the privacy threat from the network information, existing results show that an anonymous social graph can be de-anonymized by seed information (107; 69) and community structures in the social graph (111). As for the privacy threat from text information, sophisticated machine learning algorithms can be used to infer a lot of sensitive information, such as age (156; 158), location (88; 159), language (110), and political preference (31).

On the defense side, the research community only attempts to protect user privacy from the perspective of network information. The research efforts fall into two directions. The first line of research (64; 94; 129) aims at protecting vertex privacy by publishing social graphs with anonymized user IDs, and the research effort is to pre-

vent the adversary from linking anonymized IDs to corresponding real IDs in the real social network. The other line of research targets link/edge privacy, and the research effort is to publish social graphs with real user IDs but perturbed edges by publishing an obfuscated social network to protect users' privacy (103; 92). Our paper is the first to protect the privacy from the text information and is complementary to these efforts.

Some other work protects the social media users' privacy from the architectural aspect. For example, Cristofaro *et al.* proposed the Hammingbird (40) to replace the Twitter system, and Papadopoulos *et al.* proposed a K-subscription system to protect the user's privacy in Twitter (115). However, although these systems have strong privacy guarantee, they might be difficult to be widely adopted in practice.

6.5.2 Differential Privacy

Privacy-preserving data publishing has been thoroughly studied and surveyed in (54). The majority of work surveyed in (54) focuses on the traditional database instead of the unstructured social media data.

In this paper, we adopt the non-interactive differential privacy model with completeness, which has been studied in (72) for differentially-private PCA and in (89) for differentially private compressive sensing. However, the first work cannot be used for the nonnegative text analysis or the same-dimension perturbation, and the second cannot defend against the user-linkage attack. There are other non-interactive differential privacy mechanisms such as (106) and (146). However, they cannot guaranteed the completeness of the one-on-one mapping. Differential privacy has also been applied in many other applications, such as location (14; 91), compressive sensing (89), and health data (42).

6.5.3 Privacy-Preserving Machine Learning

Also related is privacy-preserving machine learning (27; 148). These schemes use a different system model where the users are distributed entities and have the control on how to share their data. We use a different system model where all the end users share their social media data with the trusted data service provider. Moreover, these schemes are designed for specific machine learning algorithms, while our framework targets more general data mining applications.

6.6 Summary

In this chapter, we investigated the vulnerability of the existing social media data publishing model and also designed the first privacy-preserving publishing framework. We pointed out that the existing ϵ-differential privacy and the popular Laplacian mechanism suffers from the curse of dimensionality and makes the perturbed data useless. We then proposed the ϵ-text indistinguishability and designed a mechanism to achieve it. We evaluated the mechanism's privacy and usefulness guarantees, as well as its high effectiveness on classification and strong defense against the user-linkage attack.

Chapter 7

CONCLUSION AND FUTURE WORK

The architectural openness and people's growing engagement continuously make the microblogging services the easy targets for the various targets. In this dissertation we uncover a number of challenging security and privacy issues in microblogging services and also propose corresponding defenses. As for the security side, we demonstrate that the *social botnet*, a group of collaborative social bots under the control of a single botmaster, can facilitate the spam distribution and digital-influence manipulation. We also propose the corresponding countermeasures and evaluate their effectiveness. Moreover, in the context of finding top-K influential microbloggers, we propose the first sybil-resilient system *TrueTop* to find the top-K influential users in microblogging services with very accurate results and strong resilience to sybil attacks. We thoroughly evaluated its performance on real-world datasets. As for the privacy side, we demonstrate that microblogging systems' structural openness and users' carelessness could disclose the later's sensitive information such as home city. We propose and evaluate *LocInfer*, a novel and lightweight system to uncover the majority of the users in any metropolitan area by exploring the geographic locality that the users in the same area have more interactions than the ones from different areas. We demonstrate one more privacy issue by proposing a novel machine learning framework *MAIF* that leverages public content and interaction information in microblogging services to infer users' hidden ages. To defend against these privacy threats, we propose the first privacy-preserving social media publishing framework to let the microblogging service providers publish their data to any third-party without disclosing users' privacy and meanwhile meeting the data's commercial utilities.

This dissertation is far from perfectness. For the future work, we can extend our dissertation in following directions.

First, Chapter 2 lists two representative attacks by social botnets and their countermeasures. Additional attacks and corresponding defenses can also be investigated. For example, the social botnets might be used to manipulate the public opinion in the crisis or significant events.

Second, Chapter 3 assumes that the sybil users have difficulty to obtain the trust from the legitimate users. Recently measurement show that the sybils have become more and more intelligent to break this constraint. How to deal with sybil's evolving intelligence is one of our future work for the TrueTop system.

Third, the LocInfer system in Chapter 4 only uses the network information to infer the location of Twitter users. As demonstrated in Chapter 5, the content information can also be combined to infer the attributes. Therefore we can exploit how to integrate the content information with the network information to infer the location of Twitter users. One possible direction is we can further refine the users in the candidate set by checking their content information.

Fourth , we can do many additional work for MAIF in Chapter 5. We seek to incorporate more interaction information such as retweets, replies, and mentions into MAIF. In addition, we will evaluate the performance of MAIF for other microblogging systems such as Tumblr. We also plan to thoroughly investigate the privacy implications of MAIF and possible countermeasures for the microbloggers particularly wary of their age privacy.

Fifth, in the privacy-preserving social media data publishing framework in Chapter 6, we will evaluate the mechanism on more applications, various attack models, extend the framework to heterogenous datasets, and investigate the (K, ϵ)-text indistinguishability mentioned in Section 6.3.7.

BIBLIOGRAPHY

[1] Klout. http://www.klout.com.

[2] Kred. http://www.kred.com/.

[3] Retweet Rank. http://www.retweetrank.com/.

[4] Jason Ding The Twitter Underground Economy: A Blooming Business. https://barracudalabs.com/2012/08/the-twitter-underground-economy-a-blooming-business/. Aug. 2012.

[5] 2013 U.S. gazetteer files. https://www.census.gov/geo/maps-data/data/gazetteer2013.html.

[6] Klout perks: A healthy two years. http://corp.klout.com/blog/2012/06/klout-perks-a-healthy-two-years/.

[7] Oauth open authentication system. http://oauth.net/.

[8] Peerindex. http://www.peerindex.com/.

[9] Twitalyzer. http://twitalyzer.com/.

[10] Twitter grader. http://twitter.grader.com/.

[11] The twitter rules. https://support.twitter.com/articles/18311-the-twitter-rules.

[12] An exhaustive study of twitter users across the world, Oct. 2012. http://www.beevolve.com/twitter-statistics/.

[13] REST API v1.1 resources. https://dev.twitter.com/docs/api/1.1/, 2013. Twitter.

[14] Miguel Andrés, Nicolás Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. Geo-indistinguishability: Differential privacy for location-based systems. In *CCS '13*, pages 901–914, Berlin, Germany, Nov. 2013.

[15] Lars Backstrom, Eric Sun, and Cameron Marlow. Find me if you can: Improving geographical prediction with social and spatial proximity. In *WWW'10*, pages 61–70, Raleigh, NC, Apr. 2010.

[16] Eytan Bakshy, Jake Hofman, Winter Mason, and Duncan Watts. Everyone's an influencer: quantifying influence on Twitter. In *WSDM'11*, pages 65–74, Hong Kong, China, Feb. 2011.

[17] Richard Baraniuk. Compressive sensing. *IEEE signal processing magazine*, 24(4), 2007.

[18] Ehrhard Behrends. *Introduction to Markov Chains With Special Emphasis on Rapid Mixing*, pages 72, 97–102. Friedrick Vieweg & Son, Oct. 2002.

[19] Richard Bellman. *Dynamic Programming*. Dover Publications, Incorporated, 2003.

[20] Fabricio Benevenuto, Gabriel Magno, Tiago Rodrigues, and Virgilio Almeida. Detecting spammers on twitter. In *CEAS'10*, Redmond, WA, July 2010.

[21] Fabrício Benevenuto, Tiago Rodrigues, Meeyoung Cha, and Virgílio Almeida. Characterizing user behavior in online social networks. In *IMC'09*, pages 49–62, Chicago, IL, Nov. 2009.

[22] Dimitri Bertsekas. *Nonlinear programming*. Athena scientific, 1999.

[23] Leyla Bilge, Thorsten Strufe, Davide Balzarotti, and Engin Kirda. All your contacts are belong to us: automated identity theft attacks on social networks. In *WWW'09*, pages 551–560, Madrid, Spain, 2009.

[24] Vincent Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[25] Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory approach to non-interactive database privacy. In *STOC '08*, pages 609–618, Victoria, British Columbia, Canada, May 2008.

[26] Yazan Boshmaf, Ildar Muslukhov, Konstantin Beznosov, and Matei Ripeanu. The socialbot network: when bots socialize for fame and money. In *ACSAC'11*, pages 93–102, Orlando, FL, Dec. 2011.

[27] Raphael Bost, Raluca Popa, Stephen Tu, and Shafi Goldwasser. Machine learning classification over encrypted data. In *NDSS'15*, San Diego, CA, Feb. 2015.

[28] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[29] Qiang Cao, Michael Sirivianos, Xiaowei Yang, and Tiago Pregueiro. Aiding the detection of fake accounts in large scale social online services. In *NSDI'12*, San Jose, CA, Apr. 2012.

[30] Meeyoung Cha, Hamed Haddadi, Fabricio Benevenuto, and Krishna Gummadi. Measuring user influence in twitter: The million follower fallacy. In *ICWSM'10*, pages 10–17, Washington, DC, May 2010.

[31] Xin Chen, Yu Wang, Eugene Agichtein, and Fusheng Wang. A comparative study of demographic attribute inference in twitter. In *ICWSM'15*, pages 590–593, Oxford, England, May 2015.

[32] Alice Cheng and Friedman Eric. Manipulability of PageRank under sybil strategies. In *NetEcon'06*, Ann Arbor, MI, June 2006.

[33] Zhiyuan Cheng, James Caverlee, and Kyumin Lee. You are where you tweet: a content-based approach to geo-locating twitter users. In *CIKM'10*, pages 759–768, Toronto, Canada, Oct. 2010.

[34] Flavio Chierichetti, Ravi Kumar, and Andrew Tomkins. Max-cover in map-reduce. In *WWW'10*, pages 231–240, Raleigh, NC, Apr. 2010.

[35] Zi Chu, Indra Widjaja, and Haining Wang. Detecting social spam campaigns on twitter. In *ACNS'12*, pages 455–475, Singapore, Jun. 2012.

[36] Hyunwoo Chun, Haewoon Kwak, Young-Ho Eom, Yong-Yeol Ahn, Sue Moon, and Hawoong Jeong. Comparison of online social relations in volume vs interaction: a case study of cyworld. In *IMC'08*, pages 57–70, Vouliagmeni, Greece, Oct. 2008.

[37] Zack Coburn and Greg Marra. Realboy: Believable twitter bots. `http://ca.olin.edu/2008/realboy/index.html`, 2008.

[38] Ryan Compton, David Jurgens, and David Allen. Geotagging one hundred million twitter accounts with total variation minimization. In *Big Data (Big Data), 2014 IEEE International Conference on*, pages 393–401, Oct. 2014.

[39] T. Cormen, C. Stein, R. Rivest, and C. Leiserson. *Introduction to Algorithms, Third Edition*. Cambridge MA: MIT Press, 2009.

[40] E. Cristofaro, C. Soriente, G. Tsudik, and A. Williams. Hummingbird: Privacy at the time of twitter. In *2012 IEEE S&P*, pages 285–299, Oakland, CA, May 2012.

[41] George Danezis and Prateek Mittal. SybilInfer: Detecting sybil nodes using social networks. In *NDSS'09*, San Diego, CA, Feb. 2009.

[42] Kamal Dankar and Khaled Emam. The application of differential privacy to health data. In *EDBT-ICDT '12*, pages 158–166, Berlin, Germany, Mar. 2012.

[43] R. Dey, Tang Cong, K. Ross, and N. Saxena. Estimating age privacy leakage in online social networks. In *INFOCOM'12*, pages 2836–2840, Orlando, FL, Mar. 2012.

[44] R. Dey, Z. Jelveh, and K. Ross. Facebook users have become much more private: A large-scale study. In *IEEE PERCOM Workshops'12*, pages 346–352, Lugano, Switzerland, March 2012.

[45] J. R. Douceur. The sybil attack. In *IPTPS '02*, Cambridge, MA, Mar. 2002.

[46] Maeve Duggan, Nicole Ellison, Cliff Lampe, Amanda Lenhart, and Mary Madden. Demographics of key social networking platforms, Jan. 2015.

[47] Cynthia Dwork. Differential privacy. In *Automata, languages and programming*, pages 1–12. Springer, 2006.

[48] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the Third Conference on Theory of Cryptography, TCC'06*, pages 265–284, New York, NY, Mar. 2006.

[49] Manuel Egele, Gianluca Stringhini, Christopher Krgel, and Giovanni Vigna. Compa: Detecting compromised accounts on social networks. In *NDSS'13*, San Diego, CA, Feb 2013.

[50] eMarketer. Us twitter user base begins to mature. `http://www.emarketer.com/Article/US-Twitter-User-Base-Begins-Mature/1010641`, Feb. 2014.

[51] Emilio Ferrara. Manipulation and abuse on social media. *SIGWEB Newsl.*, (Spring):4:1–4:9, Apr. 2015.

[52] Emilio Ferrara, Onur Varol, Clayton Davis, Filippo Menczer, and Alessandro Flammini. The rise of social bots. *arXiv preprint arXiv:1407.5225*, pages 1–11, Jul. 2014.

[53] Carlos Freitas, Fabrício Benevenuto, Saptarshi Ghosh, and Adriano Veloso. Reverse engineering socialbot infiltration strategies in twitter. *arXiv preprint arXiv:1405.4927*, pages 1–13, May 2014.

[54] Benjamin Fung, Ke Wang, Rui Chen, and Philip Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Computing Surveys (CSUR)*, 42(4):14:1–14:52, 2010.

[55] Pia Gadkari. How does twitter make money?, Nov. 2013. `http://www.bbc.com/news/business-24397472`.

[56] Hongyu Gao, Yan Chen, Kathy Lee, Diana Palsetia, and Alok Choudhary. Towards online spam filtering in social networks. In *NDSS'12*, San Diego, CA, Feb. 2012.

[57] Hongyu Gao, Jun Hu, Christo Wilson, Zhichun Li, Yan Chen, and Ben Zhao. Detecting and characterizing social spam campaigns. In *IMC'10*, pages 35–47, Melbourne, Australia, Nov. 2010.

[58] Saptarshi Ghosh, Gautam Korlam, and Niloy Ganguly. Spammers' networks within online social networks: a case-study on twitter. In *WWW'11*, pages 41–42, Bangalore, India, Mar. 2011.

[59] Saptarshi Ghosh, Bimal Viswanath, Farshad Kooti, Naveen Kumar Sharma, Korlam Gautam, Fabricio Benevenuto, Niloy Ganguly, and Krishna Gummadi. Understanding and combating link farming in the twitter social network. In *WWW'12*, pages 61–70, Lyon, France, Apr. 2012.

[60] G. Ghoshal and A. Barabasi. Ranking stability and super-stable nodes in complex networks. *Nature Communications*, 2(394), 2011.

[61] Frank Giancola. The generation gap: More myth than reality. *People and strategy*, 29(4):32, 2006.

[62] Chris Grier, Kurt Thomas, Vern Paxson, and Michael Zhang. @spam: the underground on 140 characters or less. In *CCS'10*, pages 27–37, Chicago, IL, Oct. 2010.

[63] Zoltán Gyöngyi, Hector Garcia-Molina, and Jan Pedersen. Combating web spam with trustrank. In *VLDB'04*, pages 576–587, Toronto, Canada, 2004.

[64] Michael Hay, Gerome Miklau, David Jensen, Philipp Weis, and Siddharth Srivastava. Anonymizing social networks. *Technical report, University of Massachusetts, Amherst*, page 180, 2007.

[65] Brent Hecht, Lichan Hong, Bongwon Suh, and Ed Chi. Tweets from justin bieber's heart: the dynamics of the location field in user profiles. In *CHI'11*, pages 237–246, Vancouver, Canada, May 2011.

[66] Victor Hernandez. Measuring influence online: A q&a with klout's ceo. `http://www.cnn.com/2012/02/23/tech/social-media/klout-joe-fernandez/index.html?hpt=hp\_bn6`, Feb. 2012.

[67] X. Hu, N. Sun, C. Zhang, and T. Chua. Exploiting internal and external semantics for the clustering of short texts using world knowledge. In *ACM CIKM'09*, pages 919–928, Hong Kong, China, Nov. 2009.

[68] Xia Hu, Jiliang Tang, Yanchao Zhang, and Huan Liu. Social spammer detection in microblogging. In *IJCAI 2013*, pages 2633–2639, Beijing, China, Aug. 2013.

[69] Shouling Ji, Weiqing Li, Neil Zhenqiang Gong, Prateek Mittal, and Raheem Beyah. On your social network de-anonymizablity: Quantification and large scale evaluation with seed knowledge. In *NDSS'15*, San Diago, CA, Feb. 2015.

[70] Shuiwang Ji and Jieping Ye. An accelerated gradient method for trace norm minimization. In *ICML'09*, pages 457–464, Montreal, Canada, June 2009.

[71] Jing Jiang, Christo Wilson, Xiao Wang, Peng Huang, Wenpeng Sha, Yafei Dai, and Ben Zhao. Understanding latent interactions in online social networks. *ACM Transactions on the Web (TWEB)*, 7(4):18:1–18:39, 2013.

[72] Xiaoqian Jiang, Zhanglong Ji, Shuang Wang, Noman Mohammed, Samuel Cheng, and Lucila Ohno-Machado. Differential-private data publishing through component analysis. *Transactions on data privacy*, 6(1):19, 2013.

[73] David Jurgens. That's what friends are for: Inferring location in online social media platforms based on social relationships. In *ICWSM'13*, pages 273– 282, Boston, MA, July 2013.

[74] Daniel Kifer and Ashwin Machanavajjhala. No free lunch in data privacy. In *SIGMOD'11*, pages 193–204, Athens, Greece, June 2011.

[75] Alex Knapp. A high klout score can lead to better customer service. http://www.forbes.com/sites/alexknapp/2012/06/12/a-high-klout-score-can-lead-to-better-customer-service/, Jun. 2012.

[76] D. Koll, Jun Li, J. Stein, and Xiaoming Fu. On the state of osn-based sybil defenses. In *2014 IFIP*, Trondheim, Norway, June 2014.

[77] Shamanth Kumar, Fred Morstatter, and Huan Liu. *Twitter Data Analytics*. Springer, New York, NY, USA, 2013.

[78] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. What is twitter, a social network or a news media? In *WWW'10*, pages 591–600, Raleigh, NC, May 2010.

[79] Aapo Kyrola, Guy Blelloch, and Carlos Guestrin. Graphchi: Large-scale graph computation on just a pc. In *OSDI'12*, Hollywood, CA, Oct. 2012.

[80] Amy Langville and Carl Meyer. Deeper inside pagerank. *Internet Mathematics*, 1(3):335–380, 2004.

[81] Charles Lawson and Richard Hanson. *Solving least squares problems*, volume 161. SIAM, 1974.

[82] Kyumin Lee, James Caverlee, and Steve Webb. Uncovering social spammers: social honeypots + machine learning. In *SIGIR'10*, pages 435–442, Geneva, Switzerland, July 2010.

[83] Sangho Lee and Jong Kim. WARNINGBIRD: Detecting suspicious urls in twitter stream. In *NDSS'12*, San Diego, CA, Feb. 2012.

[84] Kalev Leetaru, Shaowen Wang, Guofeng Cao, Anand Padmanabhan, and Eric Shook. Mapping the global twitter heartbeat: The geography of twitter. *First Monday*, 18(5), 2013.

[85] Jure Leskovec, Anand Rajaraman, and Jeffrey Ullman. *Mining Massive Datasets*, chapter Data Mining, pages 7–9. Cambridge University Press, 2014.

[86] Daniel Levinson. A conception of adult development. *American psychologist*, 41(1):3–13, 1986.

[87] Muyuan Li, Haojin Zhu, Zhaoyu Gao, Si Chen, Le Yu, Shangqian Hu, and Kui Ren. All your location are belong to us: Breaking mobile social networks for automated user location tracking. In *MobiHoc '14*, pages 43–52, Philadelphia, PA, Aug. 2014.

[88] Rui Li, Shengjie Wang, Hongbo Deng, Rui Wang, and Kevin Chang. Towards social user profiling: Unified and discriminative influence model for inferring home locations. In *KDD'12*, pages 1023–1031, Beijing, China, Aug. 2012.

[89] Yang Li, Zhenjie Zhang, Marianne Winslett, and Yin Yang. Compressive mechanism: utilizing sparse representation in differential privacy. In *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society*, pages 177–182, Chicago, IL, Oct. 2011.

[90] Lizi Liao, Jing Jiang, Ee-Peng Lim, and Heyan Huang. A study of age gaps between online friends. In *HT'14*, pages 98–106, Santiago, Chile, 2014.

[91] Changchang Liu, Supriyo Chakraborty, and Prateek Mittal. Dependence makes you vulnerable: Differential privacy under dependent tuples. In *NDSS'16*, San Diego, CA, Feb. 2016.

[92] Changhchang Liu and Prateek Mittal. Linkmirage: How to anonymize links in dynamic social systems. In *NDSS'16*, San Diago, CA, Feb. 2016.

[93] J. Liu, S. Ji, and J. Ye. Multi-task feature learning via efficient $l_{2,1}$-norm minimization. In *UAI'09*, June 2009.

[94] Kun Liu and Evimaria Terzi. Towards identity anonymization on graphs. In *SIGMOD'08*, pages 93–106, Vancouver, Canada, June 2008.

[95] Ryan Mac. Twitter acquires web security firm dasient. `http://www.forbes.com/sites/ryanmac/2012/01/24/twitter-acquires-web-security-firm-dasient/`, Jan. 2012.

[96] Jalal Mahmud, Jeffrey Nichols, and Clemens Drews. Where is this tweet from? inferring home locations of twitter users. In *International AAAI Conference on Weblogs and Social Media*, 2012.

[97] Jalal Mahmud, Jeffrey Nichols, and Clemens Drews. Home location identification of Twitter users. *ACM Trans. Intell. Syst. Technol.*, 5(3):47:1–47:21, Jul. 2014.

[98] Huina Mao, Xin Shuai, and Apu Kapadia. Loose tweets: An analysis of privacy leaks on Twitter. In *WPES'11*, Oct. 2011.

[99] Theodore Marinis. Psycholinguistic techniques in second language acquisition research. *Second Language Research*, 19(2):144–161, 2003.

[100] Jeffrey McGee, James Caverlee, and Zhiyuan Cheng. Location prediction in social media based on tie strength. In *CIKM '13*, pages 459–468, Burlingame, CA, Oct. 2013.

[101] Johnnatan Messias, Lucas Schmidt, Ricardo Oliveira, and Fabricio Benevenuto. You followed my bot! transforming robots into influential users in twitter. *First Monday*, 18(7), Nov. 2013.

[102] Alan Mislove, Bimal Viswanath, Krishna Gummadi, and Peter Druschel. You are who you know: inferring user profiles in online social networks. In *WSDM '10*, pages 251–260, New York city, NY, Feb. 2010.

[103] Prateek Mittal, Charalampos Papamanthou, and Dawn Song. Preserving link privacy in social network based systems. In *NDSS'13*, San Diago, CA, Feb. 2013.

[104] A. Mohaisen, H. Ttran, N. Hopeer, and Y. Kim. On the mixing time of directed social graphs and security implications. In *AsiaCCS'12*, pages 36–45, Seoul, Korea, May 2012.

[105] Abedelaziz Mohaisen, Aaram Yun, and Yongdae Kim. Measuring the mixing time of social graphs. In *IMC'10*, pages 383–389, Melbourne, Australia, Nov. 2010.

[106] Noman Mohammed, Rui Chen, Benjamin Fung, and Philip Yu. Differentially private data release for data mining. In *KDD'11*, pages 493–501, San Diego, CA, Aug. 2011.

[107] Arvind Narayanan and Vitaly Shmatikov. De-anonymizing social networks. In *SP'09*, pages 173–187, Oakland, CA, May 2009.

[108] M. Newman and M. Girvan. Analysis of weighted networks. *Physical review E*, 70(5), 2004.

[109] M. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2), 2004.

[110] Dong Nguyen, Rilana Gravel, Dolf Trieschnigg, and Theo Meder. "how old do you think i am?"; a study of language and age in twitter. In *ICWSM'13*, pages 439–448, Boston, IL, Jul. 2013.

[111] Shirin Nilizadeh, Apu Kapadia, and Yong-Yeol Ahn. Community-enhanced de-anonymization of online social networks. In *CCS '14*, pages 537–548, Scottsdale, AZ, Oct. 2014.

[112] Huseyin Oktay, Aykut Firat, and Zeynep Ertem. Demographic breakdown of twitter users: An analysis based on names. In *ASE BIGDATA/SOCIALCOM/CYBERSECURITY Conference*, Stanford University, CA, May 2014.

[113] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, Nov. 1999.

[114] A. Pal and S. Counts. Identifying topical authorities in microblogs. In *WSDM'11*, pages 45-54, Hong Kong, China, Feb. 2011.

[115] Panagiotis Papadopoulos, Antonis Papadogiannakis, Michalis Polychronakis, Apostolis Zarras, Thorsten Holz, and Evangelos P. Markatos. K-subscription: Privacy-preserving microblogging browsing through obfuscation. In *ACSAC '13*, pages 49–58, New Orleans, LA, Dec. 2013.

[116] M. Porter. *Readings in information retrieval*, chapter An algorithm for suffix stripping, pages 313–316. Morgan Kaufmann Publishers Inc., 1997.

[117] Daniele Quercia, Licia Capra, and Jon Crowcroft. The social world of Twitter: Topics, geography, and emotions. In *ICWSM'12*, pages 298–305, Dublin, Ireland, June 2012.

[118] Delip Rao, David Yarowsky, Abhishek Shreevats, and Manaswi Gupta. Classifying latent user attributes in twitter. In *SMUC'10*, Toronto, Canada, 2010.

[119] Jacob Ratkiewicz, Michael Conover, Mark Meiss, Bruno Gonçalves, Alessandro Flammini, and Filippo Menczer. Detecting and tracking political abuse in social media. In *ICWSM'11*, pages 297–304, Barcelona, Spain, July 2011.

[120] Jacob Ratkiewicz, Michael Conover, Mark Meiss, Bruno Gonçalves, Snehal Patil, Alessandro Flammini, and Filippo Menczer. Truthy: mapping the spread of astroturf in microblog streams. In *WWW'11*, pages 249–252, Hyderabad, India, Apr. 2011.

[121] Florian Schmitt and Franz Rothlauf. On the importance of the second largest eigenvalue on the convergence rate of genetic algorithms. In *GECCO'01*, San Francisco, CA, July 2001.

[122] Brian Solis. The rise of digital influence. Research report, Altimeter Group, Mar. 2012.

[123] Jonghyuk Song, Sangho Lee, and Jong Kim. Spam filtering in twitter using sender-receiver relationship. In *RAID'11*, pages 301-317, Menlo Park, CA, Sep. 2011.

[124] Vasumathi Sridharan, Vaibhav Shankar, and Minaxi Gupta. Twitter games: How successful spammers pick targets. In *ACSAC'12*, pages 389–398, Los Angeles, CA, Dec. 2012.

[125] G. Stringhini, C. Kruegel, and G. Vigna. Detecting spammers on social networks. In *ACSAC'10*, pages 1–9, Austin, TX, Dec. 2010.

[126] G. Stringhini, G. Wang, M. Egele, C. Kruegel, G. Vigna, H. Zheng, and B. Zhao. Follow the green: Growth and dynamics in twitter follower markets. In *IMC'13*, pages 163–176, Barcelona, Spain, Oct. 2013.

[127] Jingchao Sun, Rui Zhang, Xiaocong Jin, and Yanchao Zhang. Securefind: Secure and privacy-preserving object finding via mobile crowdsourcing. *CoRR*, abs/1503.07932, 2015.

[128] J. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999.

[129] Chih-Hua Tai, Peng-Jui Tseng, Philip Yu, and Ming-Syan Chen. Identities anonymization in dynamic social networks. In *ICDM'11*, pages 1224–1229, Vancouver, Canada, Dec. 2011.

[130] Jiliang Tang and Huan Liu. Unsupervised feature selection for linked social media data. In *KDD'12*, pages 904–912, Beijing, China, Aug. 2012.

[131] Kurt Thomas, Chris Grier, Justin Ma, Vern Paxson, and Dawn Song. Design and evaluation of a real-time url spam filtering service. In *IEEE S&P'11*, pages 447–462, Oakland, CA, May 2011.

[132] Kurt Thomas, Chris Grier, Vern Paxson, and Dawn Song. Suspended accounts in retrospect: An analysis of twitter spam. In *IMC'11*, pages 243–258, Berlin, Germany, Nov. 2011.

[133] Kurt Thomas, Damon McCoy, Chris Grier, Alek Kolcz, and Vern Paxson. Trafficking fraudulent accounts: The role of the underground market in twitter spam and abuse. In *USENIX Security Symposium*, Washington, DC, Aug. 2013.

[134] Robert Tibshirani. Regression shrinkage and selection via the lasso: a retrospective. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(3):273–282, 2011.

[135] Nguyen Tran, Min Bonan, Jinyang Li, and Lakshminarayanan Subramanian. Sybil-resilient online content voting. In *NSDI'09*, Boston, MA, Apr. 2009.

[136] Twitter. The get user timeline api. `https://dev.twitter.com/rest/reference/get/statuses/user_timeline`.

[137] Twitter. The search api. `https://dev.twitter.com/rest/public/search`.

[138] Bimal Viswanath, Mainack Mondal, Allen Clement, Peter Druschel, Krishna Gummadi, Alan Mislove, and Ansley Post. Exploring the design space of social network-based sybil defenses. In *COMSNETS'12*, Bangalore, India, Jan. 2012.

[139] Bimal Viswanath, Ansley Post, Krishna Gummadi, and Alan Mislove. An analysis of social network-based sybil defenses. In *SIGCOMM'10*, pages 363–374, New Delhi, India, Aug. 2010.

[140] Bimal Viswanath, Ansley Post, Krishna Gummadi, and Alan Mislove. An analysis of social network-based sybil defenses. *ACM SIGCOMM CCR*, 41(4):363–374, 2011.

[141] Alex Wang. Don't follow me - spam detection in twitter. In *SECRYPT'10*, pages 142–151, Jul. 2010.

[142] Wei Wei, Fengyuan Xu, Chiu Tan, and Qun Li. SybilDefender: Defend against sybil attacks in large social networks. In *INFOCOM'12*, pages 2492–2502, Orlando, FL, Mar. 2012.

[143] Jianshu Weng, Ee-Peng Lim, Jing Jiang, and Qi He. TwitterRank: finding topic-sensitive influential twitterers. In *WSDM'10*, pages 261–270, New York, NY, Feb. 2010.

[144] Stephen Williams. Chevy gives 3-day sonic drives to those with big klout. `http://adage.com/article/news/chevy-tests-sonics-high-klout-scores/231220/`, Nov. 2011.

[145] Christo Wilson, Bryce Boe, Alessandra Sala, Krishna Puttaswamy, and Ben Zhao. User interactions in social networks and their implications. In *EuroSys'09*, pages 205–218, Nuremberg, Germany, Apr. 2009.

[146] Xiaokui Xiao, Guozhang Wang, and Johanne Gehrke. Differential privacy via wavelet transforms. *IEEE Transactions on Knowledge and Data Engineering*, 23(8):1200–1214, 2011.

[147] Yinglian Xie, Fang Yu, Qifa Ke, Martin Abadi, Eliot Gillum, Krish Vitaldevaria, Jason Walter, Junxian Huang, and Zhuoqing Mao. Innocent by association: early recognition of legitimate users. In *CCS'12*, pages 353–364, Raleigh, NC, Oct. 2012.

[148] K. Xu, H. Yue, L. Guo, Y. Guo, and Y. Fang. Privacy-preserving machine learning algorithms for big data systems. In *ICDCS'2015*, pages 318–327, Columbus, OH, June 2015.

[149] Yuto Yamaguchi, Toshiyuki Amagasa, and Hiroyuki Kitagawa. Landmark-based user location inference in social media. In *COSN'13*, pages 223–234, Boston, MA, Oct. 2013.

[150] Chao Yang, Robert Harkreader, and Guofei Gu. Die free or live hard? empirical evaluation and new design for fighting evolving twitter spammers. In *RAID'11*, pages 318-337, Menlo Park, CA, Sep. 2011.

[151] Chao Yang, Robert Harkreader, Jialong Zhang, Suengwon Shin, and Guofei Gu. Analyzing spammers' social networks for fun and profit – a case study of cyber criminal ecosystem on twitter. In *WWW'12*, pages 71–80, Lyon, France, Apr. 2012.

[152] Chao Yang, Jialong Zhang, and Guofei Gu. A taste of tweets: Reverse engineering twitter spammers. In *ACSAC'14*, pages 86–95, New Orleans, LA, Dec. 2014.

[153] Zhi Yang, Christo Wilson, Xiao Wang, Tingting Gao, Ben Zhao, and Yafei Dai. Uncovering social network sybils in the wild. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 8(1):2, 2014.

[154] Haifeng Yu, Phillip Gibbons, Michael Kaminsky, and Feng Xiao. SybilLimit: a near-optimal social network defense against sybil attacks. *IEEE/ACM Transactions on Networking*, 18:885–898, June 2010.

[155] Haifeng Yu, Michael Kaminsky, Phillip Gibbons, and Abraham Flaxman. SybilGuard: defending against sybil attacks via social networks. In *SIGCOMM'06*, pages 267–278, Pisa, Italy, Sep. 2006.

[156] Faiyaz Zamal, Wendy Liu, and Derek Ruths. Homophily and latent attribute inference: Inferring latent attributes of twitter users from neighbors. In *ICWSM'12*, pages 287–290, Dublin, Ireland, June 2012.

[157] Chao Zhang and Vern Paxson. Detecting and analyzing automated activity on twitter. In *PAM'11*, pages 102–111, Atlanta, GA, Mar. 2011.

[158] Jinxue Zhang, Xia Hu, Yanchao Zhang, and Huan Liu. Your age is no secret: Inferring microbloggers' ages via content and interaction analysis. In *ICWSM'2016*, pages 476–485, Cologne, Germany, May 2016.

[159] Jinxue Zhang, Jingchao Sun, Rui Zhang, and Yanchao Zhang. Your actions tell where you are: Uncovering twitter users in a metropolitan area. In *CNS'15*, pages 424–432, Florence, Italy, Sep. 2015.

[160] Jinxue Zhang, Rui Zhang, Jingchao Sun, Yanchao Zhang, and Chi Zhang. Truetop: A sybil-resilient system for user influence measurement on twitter. *IEEE/ACM Trasactions on Networking*, 99(1)1–15, Octo. 2015.

[161] Jinxue Zhang, Rui Zhang, Yanchao Zhang, and Guanhua Yan. On the impact of social botnets for spam distribution and digital-influence manipulation. In *IEEE CNS'13*, pages 46–54, Washington DC, Oct. 2013.

[162] Jinxue Zhang, Rui Zhang, Yanchao Zhang, and Guanhua Yan. The rise of social botnets: Attacks and countermeasures. *IEEE Transactions on Dependable and Secure Computing*, 99(1):1–14, Apr. 2016.