Attribute Inference Attacks in Online Social Networks

NEIL ZHENQIANG GONG, Iowa State University
BIN LIU, IBM Thomas J. Watson Research Center

We propose new privacy attacks to infer attributes (e.g., locations, occupations, and interests) of online social network users. Our attacks leverage seemingly innocent user information that is publicly available in online social networks to infer missing attributes of targeted users. Given the increasing availability of (seemingly innocent) user information online, our results have serious implications for Internet privacy—private attributes can be inferred from users' publicly available data unless we take steps to protect users from such inference attacks. To infer attributes of a targeted user, existing inference attacks leverage either the user's publicly available social friends or the user's behavioral records (e.g., the web pages that the user has liked on Facebook, the apps that the user has reviewed on Google Play), but not both. As we will show, such inference attacks achieve limited success rates. However, the problem becomes *qualitatively* different if we consider both social friends and behavioral records. To address this challenge, we develop a novel model to integrate social friends and behavioral records, and design new attacks based on our model. We theoretically and experimentally demonstrate the effectiveness of our attacks. For instance, we observe that, in a real-world large-scale dataset with 1.1 million users, our attack can correctly infer *the cities a user lived in* for 57% of the users; via *confidence estimation*, we are able to increase the attack success rate to over 90% if the attacker selectively attacks half of the users. Moreover, we show that our attack can correctly infer attributes for significantly more users than previous attacks.

CCS Concepts: • **Security and privacy → Human and societal aspects of security and privacy**;

Additional Key Words and Phrases: Attribute inference, social-behavior-attribute network, privacy attack

ACM Reference format:
Neil Zhenqiang Gong and Bin Liu. 2018. Attribute Inference Attacks in Online Social Networks. *ACM Trans. Priv. Secur.* 21, 1, Article 3 (January 2018), 30 pages.
https://doi.org/10.1145/3154793

1 INTRODUCTION

Online social networks (e.g., Facebook, Google+, and Twitter) have become increasingly important platforms for users to interact with each other, process information, and diffuse social influence. A user in an online social network essentially has a list of social friends, a digital record of behaviors, and a profile. For instance, behavioral records could be a list of pages liked or shared by the user on Facebook, or they could be a set of mobile apps liked or rated by the user in Google+ or Google Play. A profile introduces the user's self-declared attributes such as majors, employers, and cities lived. To address users' privacy concerns, online social network operators provide users with

This is an extended version of Reference [22], which appeared in 2016 USENIX Security Symposium.
Authors' addresses: N. Z. Gong, Department of Electrical and Computer Engineering, Iowa State University, 307 Durham Center, Iowa State University, Ames, IA 50010; email: neilgong@iastate.edu; B. Liu, IBM Thomas J. Watson Research Center, 1101 Kitchawan Rd, Yorktown Heights, NY 10598; email: bin.liu1@ibm.com.

https://doi.org/10.1145/3154793

fine-grained privacy settings, e.g., a user could limit some attributes to be accessible only to his/her friends. Moreover, a user could also create an account without providing any attribute information. *As a result, an online social network is a mixture of both public and private user information.*

One privacy attack of increasing interest revolves around these user attributes [10, 15, 23, 27, 35, 36, 40, 42, 46, 57, 60, 64]. In this *attribute inference attack*, an attacker aims to propagate attribute information of social network users with publicly visible attributes to users with missing or incomplete attribute data. Specifically, the attacker could be any party (e.g., cyber criminal, online social network provider, advertiser, data broker, and surveillance agency) who has interests in users' private attributes. To perform such privacy attacks, the attacker only needs to collect publicly available data from online social networks. Apart from privacy risks, the inferred user attributes can also be used (by the attacker or any party who obtains the inferred attributes from the attacker) to perform various security-sensitive activities such as spear phishing [29, 54] and attacking personal-information-based backup authentication [26]. Moreover, an attacker can leverage the inferred attributes to link online users across multiple sites [2, 5, 19, 20] or with offline records (e.g., publicly available voter registration records) [45, 56] to form detailed and composite user profiles, which results in even bigger security and privacy risks.

Existing attribute inference attacks can be roughly classified into two categories, *friend-based* [15, 23, 27, 36, 40, 46, 57, 64] and *behavior-based* [10, 35, 42, 60]. Friend-based attacks are based on the intuition of *you are who you know*. Specifically, they aim to infer attributes for a user using the publicly available user attributes of the user's friends (or all other users in the social network) and the social structure among them. The foundation of friend-based attacks is *homophily*, meaning that two linked users share similar attributes [43]. For instance, if more than half of friends of a user major in computer science at a certain university, the user might also major in computer science at the same university with a high probability. Behavior-based attacks infer attributes for a user based on the public attributes of users that are similar to him/her, and the similarities between users are identified by using their behavioral data. The intuition behind behavior-based attacks is *you are how you behave*. In particular, users with the same attributes have similar interests, characteristics, and cultures so that they have similar behaviors. For instance, if a user liked apps, books, and music tracks on Google Play that are similar to those liked by users originally from China, the user might also be from China. Likewise, a previous measurement study [61] found that some apps are only popular in certain cities, implying the possibility of inferring cities a user lived in using the list of apps the user used or liked.

However, these inference attacks consider either social friendship structures or user behaviors, but not both, and thus they achieve limited inference accuracy as we will show in our experiments. Moreover, the problem of inferring user attributes becomes qualitatively different if we consider both social structures and user behaviors because features derived from them differ from each other, show different sparsity, and are at different scales. We show in our evaluation that simply concatenating features from the two sources of information regresses the overall results and reduces attack success rates.

In this article, we aim to combine social structures and user behaviors to infer user attributes. To this end, we first propose a *social-behavior-attribute* (SBA) network model to gracefully integrate social structures, user behaviors, and user attributes in a unified framework. Specifically, we add additional nodes to a social structure, each of which represents an attribute or a behavior; a link between a user and an attribute node means that the user has the corresponding attribute, and a user has a behavior if it is encoded by a link between the user and the corresponding behavior node.

Second, we design a *vote distribution attack* (VIAL) under the SBA network model to perform attribute inference. Specifically, VIAL iteratively distributes a fixed vote capacity from a *targeted user* whose attributes we want to infer to all other users in the SBA network. A user receives a

high vote capacity if the user and the targeted user are structurally similar in the SBA network, e.g., they have similar social structures and/or have performed similar behaviors. Then, each user votes for its attributes via dividing its vote capacity to them. We predict the target user to own attributes that receive the highest votes.

Third, we evaluate VIAL both theoretically and empirically, and we extensively compare VIAL with several previous attacks for inferring majors, employers, and locations using a large-scale dataset with 1.1 million users that we collected from Google+ and Google Play. For instance, we observe that our attack can correctly infer the cities a user lived in for 57% of the users; via *confidence estimation*, we are able to increase the success rate to over 90% if the attacker selectively attacks half of the users. Moreover, we find that our VIAL substantially outperforms previous attacks. Specifically, for precision, VIAL improves upon friend-based attacks and behavior-based attacks by over 20% and around 100%, respectively. These results imply that an attacker can use our attack to successfully infer private attributes of substantially more users than previous attacks.

In summary, our key contributions are as follows:

- —We propose the SBA network model to integrate social structures, user behaviors, and user attributes.
- —We design the VIAL under the SBA network model to perform attribute inference.
- —We demonstrate the effectiveness of VIAL both theoretically and empirically. Moreover, we compare VIAL with several previous attacks using a large-scale dataset that we collected from Google+ and Google Play. We observe that VIAL can correctly infer attributes for substantially more users than previous attacks.

2 PROBLEM DEFINITION AND THREAT MODEL

Attackers. The attacker could be any party who has interests in user attributes. For instance, the attacker could be a cyber criminal, online social network provider, advertiser, data broker, or surveillance agency. Cyber criminals can leverage user attributes to perform targeted social engineering attacks (now often referred to as spear phishing attacks [29, 54]) and attacking personal-information-based backup authentication [26]; online social network providers and advertisers could use the user attributes for targeted advertisements; data brokers make profit via selling the user attribute information to other parties such as advertisers, banking companies, and insurance industries [18]; and surveillance agency can use the attributes to identify users and monitor their activities.

Collecting publicly available social structures and behaviors. To perform attribute inference attacks, an attacker first needs to collect publicly available information. In particular, in our attacks, an attacker needs to collect social structures, user profiles, and user behaviors from online social networks. Such information can be collected via writing web crawlers or leveraging APIs developed by the service providers. Next, we formally describe this publicly available information.

We use an undirected[1] graph $G_s = (V_s, E_s)$ to represent a social structure, where edges in E_s represent social relationships between the nodes in V_s. We denote by $\Gamma_{u,S} = \{v|(u, v) \in E_s\}$ as the set of social neighbors of u. In addition to social network structure, we have behaviors and categorical attributes for nodes. For instance, in our Google+ and Google Play dataset, nodes are Google+ users, and edges represent friendship between users; behaviors include the set of items (e.g., apps, books, and movies) that users rated or liked on Google Play; and node attributes are derived from user profile information and include fields such as major, employer, and cities lived.

We use binary representation for user behaviors. Specifically, we treat various objects (e.g., the Android app Angry Birds, the movie *The Lord of the Rings*, and the website Facebook.com) as

[1]Our attacks can also be generalized to directed graphs.

binary variables, and we denote by m_b the total number of objects. Behaviors of a node u are then represented as a m_b-dimensional binary column vector $\vec{b}_u$ with the ith entry equal to 1 when u has performed a certain action on the ith object (*positive behavior*) and -1 when u does not perform the action on it (*negative behavior*). For instance, when we consider user review behaviors for Google+ users, objects could be items such as apps, books, and movies available in Google Play, and the action is *review*; 1 represents that the user reviewed the corresponding item and -1 means the opposite. For Facebook users, objects could be web pages; 1 represents that the user liked or shared the corresponding web page and -1 means that the user did not. We denote by $\mathbf{B} = [\vec{b}_1\ \vec{b}_2 \cdots \vec{b}_{n_s}]$ the behavior matrix for all nodes.

We distinguish between *attributes* and *attribute values*. For instance, major, employer, and location are different attributes; and each attribute could have multiple attribute values, e.g., major could be computer science, biology, or physics. A user might own a few attribute values for a single attribute. For example, a user that studies physics for undergraduate education but chooses to pursue a Ph.D. degree in computer science has two values for the attribute major. Again, we use a binary representation for each attribute value, and we denote the number of distinct attribute values as m_a. Then attribute information of a node u is represented as a m_a-dimensional binary column vector $\vec{a}_u$ with the ith entry equal to 1 when u has the ith attribute value (*positive attribute*) and -1 when u does not have it (*negative attribute*). We denote by $\mathbf{A} = [\vec{a}_1\ \vec{a}_2 \cdots \vec{a}_{n_s}]$ the attribute matrix for all nodes.

Attribute inference attacks. Roughly speaking, an attribute inference attack is to infer the attributes of a set of targeted users using the collected publicly available information. Formally, we define an attribute inference attack as follows:

Definition 2.1 (Attribute Inference Attack). Suppose we are given $T = (G_s, \mathbf{A}, \mathbf{B})$, which is a snapshot of a social network G_s with a behavior matrix $\mathbf{B}$ and an attribute matrix $\mathbf{A}$, and a list of targeted users V_t with social friends $\Gamma_{v,S}$ and binary behavior vectors $\vec{b}_v$ for all $v \in V_t$, the attribute inference attack is to infer the attribute vectors $\vec{a}_v$ for all $v \in V_t$.

We note that a user setting the friend list to be private could also be vulnerable to inference attacks. This is because the user's friends could set their friend lists publicly available. The attacker can collect a social relationship between two users if at least one of them sets the friend list to be public. Moreover, we assume the users and the service providers are not taking other steps (e.g., obfuscating social friends [28] or behaviors [13, 53, 60]) to defend against inference attacks.

Applying inferred attributes to link users across multiple online social networks and with offline records. We stress that an attacker could leverage our attribute inference attacks to further perform other attacks. For instance, a user might provide different attributes on different online social networks. Thus, an attacker could combine user attributes across multiple online social networks to better profile users, and an attacker could leverage the inferred user attributes to do so [2, 5, 19, 20]. Moreover, an attacker can further use the inferred user attributes to link online users with offline records (e.g., voter registration records) [45, 56], which results in even bigger security and privacy risks, e.g., more sophisticated social engineering attacks. We note that even if the inferred user attributes (e.g., major, employer) seem not private for some targeted users, an attacke\r could use them to link users across multiple online sites and with offline records.

3 SOCIAL-BEHAVIOR-ATTRIBUTE FRAMEWORK

We describe our *social-behavior-attribute* (SBA) network model, which integrates social structures, user behaviors, and user attributes in a unified framework. To perform our inference attacks, an

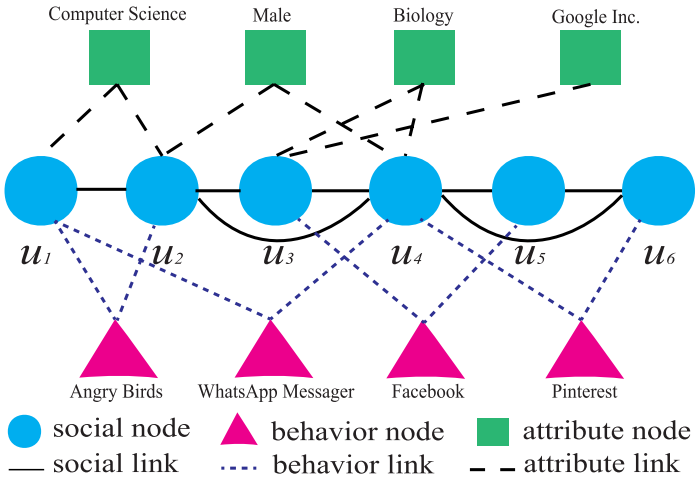


Fig. 1. Illustration of social-behavior-attribute network.

attacker needs to construct an SBA network to represent his/her collected publicly available social structures, user attributes, and behaviors.

Given a social network $G_s = (V_s, E_s)$ with m_b behavior objects, a behavior matrix $\mathbf{B}$, m_a distinct attribute values, and an attribute matrix $\mathbf{A}$, we create an augmented network by adding m_b additional nodes to G_s, with each node corresponding to a behavior object, and another m_a additional nodes to G_s, with each additional node corresponding to an attribute value. For each node u in G_s with positive attribute a or positive behavior b, we create an undirected link between u and the additional node corresponding to a or b in the augmented network. Moreover, we add the targeted users into the augmented network by connecting them to their friends and the additional nodes corresponding to their positive behaviors. We call this augmented network *social-behavior-attribute* (SBA) network since it integrates the interactions among social structures, user behaviors, and user attributes. We note that the SBA network model extends our social-attribute-network (SAN) model [23] by further including behavior information into the framework.

Nodes in the SBA framework corresponding to nodes in G_s or targeted users in V_t are called *social nodes*, nodes representing behavior objects are called *behavior nodes*, and nodes representing attribute values are called *attribute nodes*. Moreover, we use S, B, and A to represent the three types of nodes, respectively. Links between social nodes are called *social links*, links between social nodes and behavior nodes are called *behavior links*, and links between social nodes and attribute nodes are called *attribute links*. Note that there are no links between behavior nodes and attribute nodes. Figure 1 illustrates an example SBA network, in which the two social nodes u_5 and u_6 correspond to two targeted users. The behavior nodes in this example correspond to Android apps, and a behavior link represents that the corresponding user used the corresponding app. Intuitively, the SBA framework explicitly describes the sharing of behaviors and attributes across social nodes. Moreover, under the SBA framework, attribute inference involves predicting attribute links.

We also place weights on various links in the SBA framework. These link weights balance the influence of social links versus behavior links versus attribute links.[2] For instance, weights on social links could represent the tie strengths between social nodes. Users with stronger tie strengths

[2]In principle, we could also assign weights to nodes to incorporate their relative importance. However, our attack does not rely on node weights, so we do not discuss them.

could be more likely to share the same attribute values. The weight on a behavior link could indicate the predictiveness of the behavior in terms of the user's attributes. In other words, a behavior link with a higher weight means that performing the corresponding behavior better predicts the attributes of the user. For instance, if we want to predict user gender, the weight of the link between a female user and a mobile app tracking women's monthly periods could be larger than the weight of the link between a male user and the app. Weights on attribute links can represent the degree of affinity between users and attribute values. For instance, an attribute link connecting the user's hometown could have a higher weight than the attribute link connecting a city where the user once travelled. We discuss how link weights can be learnt via machine learning in Section 8.

We denote an SBA network as $G = (V, E, w, t)$, where V is the set of nodes, $n = |V|$ is the total number of nodes, E is the set of links, $m = |E|$ is the total number of links, w is a function that maps a link to its link weight, i.e., w_{uv} is the weight of link (u, v), and t a function that maps a node to its node type, i.e., t_u is the node type of u. For instance, $t_u = S$ means that u is a social node. Additionally, for a given node u in the SBA network, we denote by Γ_u, $\Gamma_{u,S}$, $\Gamma_{u,B}$, and $\Gamma_{u,A}$, respectively, the sets of *all neighbors*, *social neighbors*, *behavior neighbors*, and *attribute neighbors* of u. Moreover, for links that are incident from u, we use d_u, $d_{u,S}$, $d_{u,B}$, and $d_{u,A}$ to denote the sum of weights of all links, weights of links connecting social neighbors, weights of links connecting behavior neighbors, and weights of links connecting attribute neighbors, respectively. More specifically, we have $d_u = \sum_{v \in \Gamma_u} w_{uv}$ and $d_{u,Y} = \sum_{v \in \Gamma_{u,Y}} w_{uv}$, where $Y = S, B, A$.

Furthermore, we define two types of *hop-2 social neighbors* of a social node u, which share common behavior neighbors or attribute neighbors with u. In particular, a social node v is called a *behavior-sharing* social neighbor of u if v and u share at least one common behavior neighbor. For instance, in Figure 1, both u_2 and u_4 are behavior-sharing social neighbors of u_1. We denote the set of behavior-sharing social neighbors of u as $\Gamma_{u,BS}$. Similarly, we denote the set of attribute-sharing social neighbors of u as $\Gamma_{u,AS}$. Formally, we have $\Gamma_{u,BS}=\{v|t(v) = S \ \& \ \Gamma_{v,B} \cap \Gamma_{u,B} \neq \emptyset\}$ and $\Gamma_{u,AS}=\{v|t(v) = S \ \& \ \Gamma_{v,A} \cap \Gamma_{u,A} \neq \emptyset\}$. We note that our definitions of $\Gamma_{u,BS}$ and $\Gamma_{u,AS}$ also include the social node u itself. These notations will be useful in describing our attack.

4 VOTE DISTRIBUTION ATTACK (VIAL)

4.1 Overview

Suppose we are given an SBA network G that also includes the social structures and behaviors of the targeted users; our goal is to infer attributes for every targeted user. Specifically, for each targeted user v, we compute the similarity between v and each attribute value (corresponding to an attribute node in G), and then we predict that v owns the attribute values that have the highest similarity scores.

In a high-level abstraction, VIAL works in two phases.

- *Phase I.* VIAL iteratively distributes a fixed vote capacity from the targeted user v to the rest of users in Phase I. The intuitions are that a user receives a high vote capacity if the user and the targeted user are structurally similar in the SBA network (e.g., share common friends and behaviors), and that the targeted user is more likely to have the attribute values belonging to users with higher vote capacities. At the end of Phase I, we obtain a vote capacity vector $\vec{s}_v$, where $\vec{s}_{vu}$ is the vote capacity of the social node u.
- *Phase II.* Intuitively, if a user with a certain vote capacity has more attribute values, then, according to the information of this user alone, the likelihood of each of these attribute values belonging to the targeted user decreases. Moreover, an attribute value should receive more votes if more users with higher vote capacities have the attribute value. Therefore, in

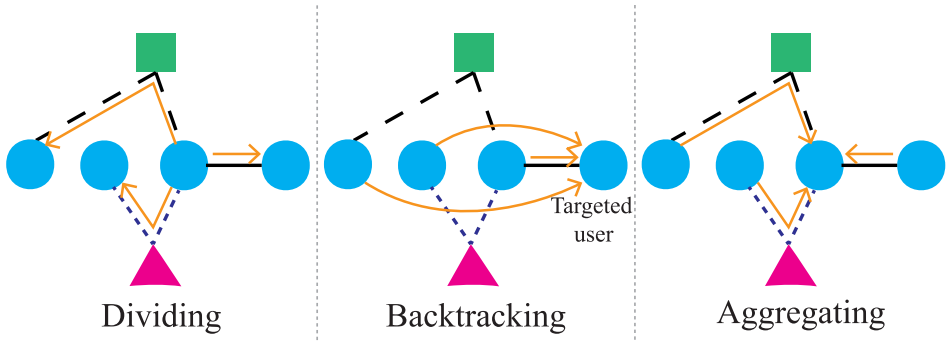


Fig. 2. Illustration of our three local rules.

Phase II, each social node votes for its attribute values via dividing its vote capacity among them, and each attribute value sums the vote capacities that are divided to it by its social neighbors. We treat the summed vote capacity of an attribute value as its similarity with v. Finally, we predict v has the attribute values that receive the highest votes.

4.2 Phase I

In Phase I, VIAL iteratively distributes a fixed vote capacity from the targeted user v to the rest of users. We denote by $\vec{s}_v^{(i)}$ the vote capacity vector in the ith iteration, where $\vec{s}_{vu}^{(i)}$ is the vote capacity of node u in the ith iteration. Initially, v has a vote capacity $|V_s|$ and all other social nodes have vote capacities of 0. Formally, we have

$$\vec{s}_{vu}^{(0)} = \begin{cases} |V_s| & \text{if } u = v \\ 0 & \text{otherwise} \end{cases}. \tag{1}$$

In each iteration, VIAL applies three local rules. They are *dividing*, *backtracking*, and *aggregating*. Intuitively, if a user u has more (hop-2) social neighbors, then each neighbor could receive less vote capacity from u. Therefore, our dividing rule splits a social node u's vote capacity to u's social neighbors and hop-2 social neighbors. The backtracking rule takes a portion of every social node's vote capacity and assigns them back to the targeted user v, which is based on the intuition that social nodes that are closer to v in the SBA network are likely to be more similar to v and should get more vote capacities. A user could have a higher vote capacity if it is linked to more social neighbors and hop-2 social neighbors with higher vote capacities. Thus, for each user u, the aggregating rule collects the vote capacities that are shared to u by its social neighbors and hop-2 social neighbors in the dividing step. Figure 2 illustrates the three local rules. Next, we discuss the three local rules in detail.

Our aggregating rule to compute the new vote capacity $\vec{s}_{vu}^{(i)}$ for u:

$$\vec{s}_{vu}^{(i)} = \begin{cases} (1-\alpha)(\sum_{x\in\Gamma_{u,S}} p_v^{(i)}(x,u) + \sum_{x\in\Gamma_{u,BS}} q_v^{(i)}(x,u) + \sum_{x\in\Gamma_{u,AS}} r_v^{(i)}(x,u)) & \text{if } u \neq v \\ (1-\alpha)(\sum_{x\in\Gamma_{u,S}} p_v^{(i)}(x,u) + \sum_{x\in\Gamma_{u,BS}} q_v^{(i)}(x,u) + \sum_{x\in\Gamma_{u,AS}} r_v^{(i)}(x,u)) + \alpha|V_s| & \text{otherwise} \end{cases}. \tag{2}$$

Our dividing matrix:

$$\mathbf{M}_{ux} = \begin{cases} \delta_{ux,S} \cdot \frac{w_S}{w_T} \cdot \frac{w_{ux}}{d_{u,S}} + \delta_{ux,BS} \cdot \frac{w_{BS}}{w_T} \cdot w_B(u,x) + \delta_{ux,AS} \cdot \frac{w_{AS}}{w_T} \cdot w_A(u,x) & \text{if } x \in \gamma_u \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

where $\gamma_u = \Gamma_{u,S} \cup \Gamma_{u,BS} \cup \Gamma_{u,AS}$; $\delta_{ux,Y} = 1$ if $x \in \Gamma_{u,Y}$, otherwise $\delta_{ux,Y} = 0$, $Y = S, BS, AS$.

Dividing. A social node u could have social neighbors, behavior-sharing social neighbors, and attribute-sharing social neighbors. To distinguish them, we use three weights, w_S, w_{BS}, and w_{AS}, to represent the shares of them, respectively. For instance, the total vote capacity shared to social neighbors of u in the tth iteration is $\vec{s}_{vu}^{(i-1)} \times \frac{w_S}{w_S+w_{BS}+w_{AS}}$. Then we further divide the vote capacity among each type of neighbors according to their link weights. We define $I_{u,Y} = 1$ if the set of neighbors $\Gamma_{u,Y}$ is non-empty, otherwise $I_{u,Y} = 0$, where $Y = S, BS, AS$. The variables $I_{u,S}$, $I_{u,BS}$, and $I_{u,AS}$ are used to consider the scenarios where u does not have some type(s) of neighbors, in which u's vote capacity is divided among less than three types of social neighbors. For convenience, we denote $w_T = w_S I_{u,S} + w_{BS} I_{u,BS} + w_{AS} I_{u,AS}$.

— *Social neighbors.* A social neighbor $x \in \Gamma_{u,S}$ receives a higher vote capacity from u if their link weight (e.g., tie strength) is higher. Therefore, we model the vote capacity $p_v^{(i)}(u,x)$ that is divided to x by u in the ith iteration as

$$p_v^{(i)}(u,x) = \vec{s}_{vu}^{(i-1)} \cdot \frac{w_S}{w_T} \cdot \frac{w_{ux}}{d_{u,S}}, \tag{4}$$

where $d_{u,S}$ is the summation of weights of social links that are incident from u.

— *Behavior-sharing social neighbors.* A behavior-sharing social neighbor $x \in \Gamma_{u,BS}$ receives a higher vote capacity from u if they share more behavior neighbors with higher predictiveness. Thus, we model vote capacity $q_v^{(i)}(u,x)$ that is divided to x by u in the ith iteration as

$$q_v^{(i)}(u,x) = \vec{s}_{vu}^{(i-1)} \cdot \frac{w_{BS}}{w_T} \cdot w_B(u,x), \tag{5}$$

where $w_B(u,x) = \sum_{y \in \Gamma_{u,B} \cap \Gamma_{x,B}} \frac{w_{uy}}{d_{u,B}} \cdot \frac{w_{xy}}{d_{y,S}}$, representing the overall share of vote capacity that u divides to x because of their common behavior neighbors. Specifically, $\frac{w_{uy}}{d_{u,B}}$ characterizes the fraction of vote capacity, u divides to the behavior neighbor y, and $\frac{w_{xy}}{d_{y,S}}$ characterizes the fraction of vote capacity y divides to x. Large weights of w_{uy} and w_{xy} indicate y is a predictive behavior of the attribute values of u and x, and having more such common behavior neighbors make x share more vote capacity from u.

— *Attribute-sharing social neighbors.* An attribute-sharing social neighbor $x \in \Gamma_{u,AS}$ receives a higher vote capacity from u if they share more attribute neighbors with higher degree of affinity. Thus, we model vote capacity $r_v^{(i)}(u,x)$ that is divided to x by u in the ith iteration as

$$r_v^{(i)}(u,x) = \vec{s}_{vu}^{(i-1)} \cdot \frac{w_{AS}}{w_T} \cdot w_A(u,x), \tag{6}$$

where $w_A(u,x) = \sum_{y \in \Gamma_{u,A} \cap \Gamma_{x,A}} \frac{w_{uy}}{d_{u,A}} \cdot \frac{w_{xy}}{d_{y,S}}$, representing the overall share of vote capacity that u divides to x because of their common attribute neighbors. Specifically, $\frac{w_{uy}}{d_{u,A}}$ characterizes the fraction of vote capacity u divides to the attribute neighbor y and $\frac{w_{xy}}{d_{y,S}}$ characterizes the fraction of vote capacity y divides to x. Large weights of w_{uy} and w_{xy} indicate y is an attribute value with a high degree of affinity, and having more such common attribute values makes x share more vote capacity from u.

We note that a social node x could be multiple types of social neighbors of u (e.g., x could be social neighbor and behavior-sharing social neighbor of u), in which x receives multiple shares of vote capacity from u and we sum them as x's final share of vote capacity.

Backtracking. For each social node u, the backtracking rule takes a portion α of u's vote capacity back to the targeted user v. Specifically, the vote capacity backtracked to v from u is $\alpha \vec{s}_{vu}^{(i-1)}$.

ALGORITHM 1: Phase I of VIAL

Input: $G = (V, E, w, t)$, $\mathbf{M}$, v, ϵ, and α.
Output: $\vec{s}_v$.
begin
  //Initializing the vote capacity vector.
  for $u \in V_s$ **do**
    if $u = v$ **then**
      $\vec{s}_{vu}^{(0)} \longleftarrow |V_s|$
    else
      $\vec{s}_{vu}^{(0)} \longleftarrow 0$
    end
  end
  $error \longleftarrow 1$
  while $error > \epsilon$ **do**
    $\vec{s}_v^{(i)} \longleftarrow \alpha\vec{e}_v + (1-\alpha)\mathbf{M}^T\vec{s}_v^{(i-1)}$
    $error \longleftarrow |\vec{s}_v^{(i)} - \vec{s}_v^{(i-1)}|/|V_s|$
  end
  return $\vec{s}_v^{(i)}$
end

Considering backtracking, the vote capacity divided to the social neighbor x of u in the dividing step is modified as $(1-\alpha)p_v^{(i)}(u,x)$. Similarly, the vote capacities divided to a behavior-sharing social neighbor and an attribute-sharing social neighbor x are modified as $(1-\alpha)q_v^{(i)}(u,x)$ and $(1-\alpha)r_v^{(i)}(u,x)$, respectively. We call the parameter α *backtracking strength*. A larger backtracking strength enforces more vote capacity to be distributed among the social nodes that are closer to v in the SBA network. $\alpha = 0$ means no backtracking. We will show that, via both theoretical and empirical evaluations, VIAL achieves better inference accuracy when using backtracking.

We can verify that $\sum_{x\in\Gamma_{u,S}} \frac{w_{ux}}{d_{u,S}} = 1$, $\sum_{x\in\Gamma_{u,BS}} w_B(u,x) = 1$, and $\sum_{x\in\Gamma_{u,AS}} w_A(u,x) = 1$ for every user u in the dividing step. In other words, every user divides all its vote capacity to its neighbors (including the user itself if the user has hop-2 social neighbors). Therefore, the total vote capacity keeps unchanged in every iteration. Thus, the total vote capacity that is backtracked to the targeted user is $\alpha|V_s|$.

Aggregating. The aggregating rule computes a new vote capacity $\vec{s}_{vu}^{(i)}$ for u by aggregating the vote capacities that are divided to u by its neighbors in the ith iteration. For the targeted user v, we also collect the vote capacities that are backtracked from all social nodes. Formally, our aggregating rule is represented as Equation (2).

Matrix representation. We derive Phase I of our attack using matrix terminologies, which makes it easier to iteratively compute the vote capacities. Towards this end, we define a *dividing matrix* $\mathbf{M} \in R^{|V_s|\times|V_s|}$, which is formally represented in Equation (3). The dividing matrix encodes the dividing rule. Specifically, u divides $\mathbf{M}_{ux}$ fraction of its vote capacity to the neighbor x in the dividing step. Note that $\mathbf{M}$ includes the dividing rule for all three types of social neighbors.

With the dividing matrix $\mathbf{M}$, we can represent the backtracking and aggregating rules in the ith iteration as follows:

$$\vec{s}_v^{(i)} = \alpha\vec{e}_v + (1-\alpha)\mathbf{M}^T\vec{s}_v^{(i-1)}, \tag{7}$$

where $\vec{e}_v$ is a vector with the vth entry equals $|V_s|$ and all other entries equal 0, and $\mathbf{M}^T$ is the transpose of $\mathbf{M}$.

Given an initial vote capacity vector specified in Equation (1), we iteratively apply Equation (7) until the difference between the vectors in two consecutive iterations is smaller than a predefined threshold. Algorithm 1 shows Phase I of our attack.

4.3 Phase II

In Phase I, we obtained a vote capacity for each user. On one hand, the targeted user could be more likely to share attribute values with the users with higher vote capacities. On the other hand, if a user has more attribute values, then the likelihood of each of these attribute values belonging to the targeted user could be smaller. For instance, if a user with a high vote capacity once studied in more universities for undergraduate education, then according to this user's information alone, the likelihood of the targeted user studying in each of those universities could be smaller.

Moreover, among a user's attribute values, an attribute value that has a higher degree of affinity (represented by the weight of the corresponding attribute link) with the user could be more likely to be an attribute value of the targeted user. For instance, suppose a user once lived in two cities, one of which is the user's hometown while the other is a city where the user once travelled; the user has a high vote capacity because he/she is structurally close (e.g., he/she shares many common friends with the targeted user) to the targeted user; then the targeted user is more likely to be from the hometown of the user than from the city where the user once travelled.

Therefore, to capture these observations, we divide the vote capacity of each user to its attribute values in proportion to the weights of the corresponding attribute links; and each attribute value sums the vote capacities that are divided to it by the users having the attribute value. Intuitively, an attribute value receives more votes if more users with higher vote capacities link to the attribute value via links with higher weights. Formally, we have

$$\vec{t}_{va} = \sum_{u \in \Gamma_{a,S}} \vec{s}_{vu} \cdot \frac{w_{au}}{d_{u,A}}, \tag{8}$$

where $\vec{t}_{va}$ is the final votes of the attribute value a, $\Gamma_{a,S}$ is the set of users who have the attribute value a, $d_{u,A}$ is the sum of weights of attribute links that are incident from u.

We treat the summed votes of an attribute value as its similarity with v. Finally, we predict v has the attribute values that receive the highest votes.

4.4 Confidence Estimation

For a targeted user, a *confidence estimator* takes the final votes for all attribute values as an input and produces a confidence score. A higher confidence score means that attribute inference for the targeted user is more trustworthy. We design a confidence estimator based on clustering techniques. A targeted user could have multiple attribute values for a single attribute, and our attack could produce close votes for these attribute values. Therefore, we design a confidence estimator called *clusterness* for our attack. Specifically, we first use a clustering algorithm (e.g., k-means [33]) to group the votes that our attack produces for all candidate attribute values into two clusters. Then we compute the average vote in each cluster, and the clusterness is the difference between the two average votes. The intuition of our clusterness is that if our attack successfully infers the targeted user's attribute values, there could be a cluster of attribute values whose votes are significantly higher than other attribute values'.

Suppose the attacker chooses a confidence threshold and only predicts attributes for targeted users whose confidence scores are higher than the threshold. Via setting a larger confidence threshold, the attacker will attack less targeted users but could achieve a higher success rate. In other

words, an attacker can balance between the success rates and the number of targeted users to attack via confidence estimation.

5 THEORETICAL ANALYSIS

We analyze the convergence of VIAL and derive the analytical forms of vote capacity vectors, discuss the importance of the backtracking rule, and analyze the complexity of VIAL.

5.1 Convergence and Analytical Solutions

We first show that for any backtracking strength $\alpha \in (0, 1]$, the vote capacity vectors converge.

Theorem 5.1. *For any backtracking strength $\alpha \in (0, 1]$, the vote capacity vectors $\vec{s}_v^{(0)}, \vec{s}_v^{(1)}, \vec{s}_v^{(2)}, \cdots$ converge, and the converged vote capacity vector is $\alpha(I - (1-\alpha)\mathbf{M}^T)^{-1}\vec{e}_v$. Formally, we have*

$$\vec{s}_v = \lim_{i\to\infty} \vec{s}_v^{(i)} = \alpha(I - (1-\alpha)\mathbf{M}^T)^{-1}\vec{e}_v, \tag{9}$$

where I is an identity matrix and $(I - (1-\alpha)\mathbf{M}^T)^{-1}$ is the inverse of $(I - (1-\alpha)\mathbf{M}^T)$.

Proof. See Appendix A. □

Next, we analyze the convergence of VIAL and the analytical form of the vote capacity vector when the backtracking strength $\alpha = 0$.

Theorem 5.2. *When $\alpha = 0$ and the SBA network is connected, the vote capacity vectors $\vec{s}_v^{(0)}, \vec{s}_v^{(1)}, \vec{s}_v^{(2)}, \cdots$ converge, and the converged vote capacity vector is proportional to the unique stationary distribution of the Markov chain whose transition matrix is $\mathbf{M}$. Mathematically, the converged vote capacity vector $\vec{s}_v$ can be represented as*

$$\vec{s}_v = |V_s|\vec{\pi}, \tag{10}$$

where $\vec{\pi}$ is the unique stationary distribution of the Markov chain whose transition matrix is $\mathbf{M}$.

Proof. See Appendix B. □

With Theorem 5.2, we have the following corollary, which states that the vote capacity of a user is proportional to its weighted degree for certain assignments of the shares of social neighbors and hop-2 social neighbors in the dividing step.

Corollary 5.1. *When $\alpha = 0$, the SBA network is connected, and for each user u, the shares of social neighbors, behavior-sharing social neighbors, and attribute-sharing social neighbors in the dividing step are $w_S = \tau \cdot d_{u,S}$, $w_{BS} = \tau \cdot d_{u,B}$, and $w_{AS} = \tau \cdot d_{u,A}$, respectively, then we have*

$$\vec{s}_{vu} = |V_s|\frac{d_u}{D}, \tag{11}$$

where τ is any positive number, d_u is the weights of all links of u, and D is the twice of the total weights of all links in the SBA network, i.e., $D = \sum_u d_u$.

Proof. See Appendix C. □

5.2 Importance of Backtracking

Theorem 5.2 implies that when there is no backtracking, the converged vote capacity vector is independent with the targeted users. In other words, VIAL with no backtracking predicts the same attribute values for all targeted users. This explains why VIAL with no backtracking achieves suboptimal performance. We will further empirically evaluate the impact of backtracking strength in our experiments. We found that VIAL's performance significantly degrades when there is no backtracking.

5.3 Time Complexity

The major cost of VIAL is from Phase I, which includes computing $\mathbf{M}$ and iteratively computing the vote capacity vector. $\mathbf{M}$ only needs to be computed once and is applied to all targeted users. $\mathbf{M}$ is a sparse matrix with $O(m)$ non-zero entries, where m is the number of links in the SBA network. To compute $\mathbf{M}$, for every social node, we need to go through its social neighbors and hop-2 social neighbors; for a hop-2 social neighbor, we need to go through the common attribute/behavior neighbors between the social node and the hop-2 social neighbor. Therefore, the time complexity of computing $\mathbf{M}$ is $O(m)$.

Using sparse matrix representation of $\mathbf{M}$, the time complexity of each iteration (i.e., applying Equation (7) in computing the vote capacity vector) is $O(m)$. Therefore, the time complexity of computing the vote capacity vector for one targeted user is $O(d \cdot m)$, where d is the number of iterations. Thus, the overall time complexity of VIAL is $O(d \cdot m)$ for one targeted user.

6 DATA COLLECTION

We collected a dataset from Google+ and Google Play to evaluate our VIAL and previous attacks. Specifically, we collected social structures and user attributes from Google+ and user review behaviors from Google Play. Google assigns each user a 21-digit universal ID, which is used in both Google+ and Google Play. We first collected a social network with user attributes from Google+ via iteratively crawling users' friends. Then we crawled review data of users in the Google+ dataset. All the information that we collected is publicly available. To evaluate various attribute inference attacks, we will randomly sample some users with publicly available attributes, treat them as targeted users, and remove their attributes as ground truth.

6.1 Google+ Dataset

Each user in Google+ has an outgoing friend list (i.e., "in your circles"), an incoming friend list (i.e., "have you in circles"), and a profile. Shortly after Google+ was launched in late June 2011, Gong et al. [23, 24] began to crawl daily snapshots of public Google+ social network structure and user profiles (e.g., *major*, *employer*, and *cities lived*). Their dataset includes 79 snapshots of Google+ collected from July 6 to October 11, 2011. Each snapshot was a large Weakly Connected Component of Google+ social network at the time of crawling.

Gong et al. made a part of this dynamic Google+ dataset publicly available. We obtained one snapshot from Gong et al. To better approximate friendships between users, we construct an undirected social network from the crawled Google+ dataset via keeping an undirected link between a user u and v if u is in v's both incoming friend list and outgoing friend list. After preprocessing, our Google+ dataset consists of 1,111,905 users and 5,328,308 undirected social links.

6.2 Crawling Google Play

The Google Play market [25] is a centralized platform where developers can publish their *items* while users can download and consume them. There are seven categories of items in Google Play. They are *apps, tv, movies, music, books, newsstand,* and *devices*.

Google Play also provides two review mechanisms for users to provide feedback on an item. They are the *liking* mechanism and the *rating* mechanism. In the liking mechanism, a user simply clicks a like button to express his preference about an item. In the rating mechanism, a user is able to express fine-grained preferences. Specifically, a user gives a rating score which is in the set {1,2,3,4,5} as well as a detailed comment to support his/her rating. A score of 1 represents low preference and a score of 5 represents high preference. We say a user *reviewed* an item if the user rated or liked the item.

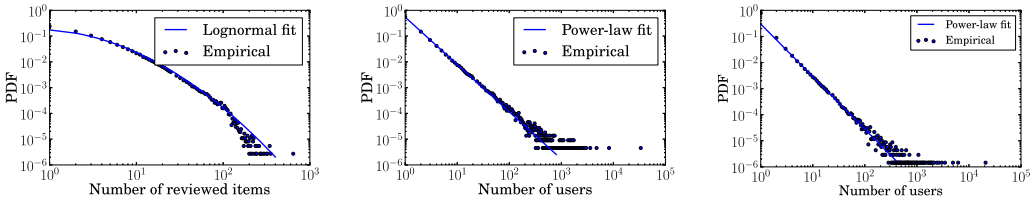


Fig. 3. (a) The probability distribution of the number of items that are reviewed by a user. (b) The probability distribution of the number of users that reviewed an item. (c) The probability distribution of the number of users that have a certain attribute value.

Users' reviews are publicly available in Google Play. Specifically, after a user u logs into Google Play, u can view the list of items reviewed by any user v once u can obtain v's Google ID.[3] We crawled the list of items reviewed by each user in the Google+ dataset. In total, we collected 260,245 items and 3,954,822 reviews.

6.3 Basic Measurement

6.3.1 User Reviews. We find that 33% of users have reviewed at least one item. Figure 3(a) further shows the probability distribution of the number of items that were reviewed by these users. We observe that this distribution is best fit as a *lognormal* distribution, i.e., a large fraction of users reviewed a small number of items and there are a tiny fraction of users that reviewed a large number of items. For instance, 86% of users reviewed less than 5 items and there are still 1% of users that reviewed at least 50 items. Our observations show that attribute inference using review data alone is limited to a small number of users. However, we will show—in our experiments—that once we combine users' review data, VIAL achieves better attack success rates. Moreover, we will show that as users review more items, our attack can achieve better attack success rates.

Figure 3(b) demonstrates the probability distribution of the number of users that reviewed an item. We find that this distribution is best fit as a *power-law* distribution, i.e., a large fraction of items were reviewed by a small number of users and there are a small fraction of items that were reviewed by a large number of users. Since items with too few reviews might not be informative to distinguish users with different attribute values, we use items that were reviewed by at least five users. After preprocessing, we have 48,706 items and 3,635,231 reviews.

6.3.2 User Attributes. We consider three attributes, i.e., *major*, *employer*, and *cities lived*. We note that, although we focus on these attributes that are available to us at a large scale, our attack is also applicable to infer other attributes such as sexual orientation, political views, and religious views. Moreover, some targeted users might not view inferring these attributes as a privacy attack, but an attacker can leverage these attributes to further link users across online social networks [2, 5, 19, 20] or even link them with offline records to perform more serious security and privacy attacks [45, 56].

We take the strings input by a user in its Google+ profile as attribute values. Figure 3(c) shows the probability distribution of the number of users that have a given attribute value. Again, we observe a *power-law* distribution, i.e., most attribute values are owned by a small number of users, and there are some attribute values that are owned by a large number of users. Users could fill in their profiles freely in Google+, which could be one reason that we observe many infrequent attribute values. Specifically, different users might have different names for the same attribute

[3]For instance, https://play.google.com/store/people/details?id=109649490051533913252 shows the list of items reviewed by the user with a Google ID 109649490051533913252.

Table 1. Sixty-Two Distinct Majors That Are Classified into Five Categories

Category	Examples	#majors	% of users
Arts, Humanities, Social Sciences	Music, History, Sociology	13	1.3%
Information Technology	CS, ECE, EE	14	3.6%
Business	Marketing, Accounting, MBA	8	1.2%
Science, Math, and Engineering	Biology, Mathematics Mechanical Engineering	17	1.8%
Public and Social Services	Law, Nursing, Public Relations	10	0.5%

Table 2. Seventy-Eight Distinct Employers That Are Classified into Six Categories

Category	Examples	#employers	% of users
Public and Social Services	U. S. Army, U.S. Navy, UPS	5	0.1%
Universities	UC Berkeley, Harvard University Princeton University	20	0.8%
Trades and Personal Services	Best Buy, Starbucks, Subway	10	0.2%
Science, Math, and Engineering	Boeing, Ford, General Motors	3	0.1%
Information Technology	Google, IBM, Microsoft	29	1.5%
Finance	Bank of America, Chase, Goldman Sachs	11	0.4%

value. For instance, the major of computer science could also be abbreviated as CS by some users. Indeed, we find that 20,861 users have computer science as their major and 556 users have CS as their major in our dataset. Moreover, small typos (e.g., one letter is incorrect) in the free-form inputs make the same attribute value be treated as a different one.

Major. We consider the top 100 majors that are claimed by the most users. We manually merge the majors that actually refer to the same one, e.g., computer science and CS, Btech and biotechnology. Moreover, some majors are close to each other in the sense that users with those majors might have similar characteristics, behaviors, and interests, e.g., computer science vs. computer engineering. Since it is unnecessary to distinguish such majors in some applications such as inferring hidden social relationships between users and targeted advertisements, we further classify majors into categories. In particular, we first get five major categories constructed by the organization Bigfuture [8], which is widely used by students to guide the selection of majors and careers. Then we manually analyze the category of each major that we consider. Table 1 shows the five categories of majors, the number of distinct majors in each category, and the fraction of users in our dataset that have at least one major in a given category. We note that the fractions of users are small because we consider a small number of top majors and only around 30% of users in our dataset have at least one publicly available attribute value.

Employer. Similar to major, we select the top 100 employers that are claimed by the most users, manually merge the employers that refer to the same one, and classify them into six categories. Table 2 shows the six categories, the number of distinct employers in each category, and the fraction of users that have at least one employer in a given category.

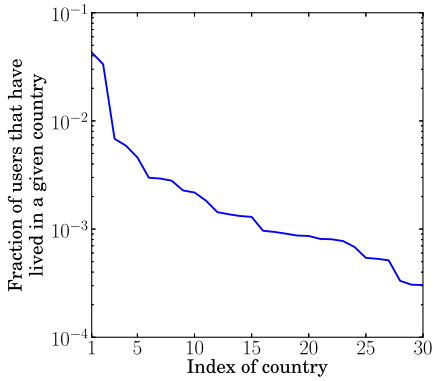


Fig. 4. Fraction of users that have lived in a given country.

Cities and countries lived. Again, we select the top 100 cities in which most users in the Google+ dataset claimed they have lived in. We first manually merge the cities that actually refer to the same one. Then, we map cities to countries. After preprocessing, we have 70 distinct cities and 30 distinct countries including the U.S., India, and China. Figure 4 shows the fraction of users that have lived in a given country.

Summary. We consider 210 distinct attribute values, including 62 majors, 78 employers, and 70 cities. Moreover, we classify these attribute values into 5 categories of majors, 6 categories of employers, and 30 countries. We acknowledge that our Google+ dataset might not be a representative sample of the recent entire Google+ social network, and thus the inference attack success rates obtained in our experiments might not represent those of the entire Google+ social network.

6.4 Constructing SBA Networks

We take each user in the Google+ dataset as a social node and links between them as social links. For each item in our Google Play dataset, we add a corresponding behavior node. If a user reviewed an item, we create a link between the corresponding social node and the corresponding behavior node. That a user reviewed an item means that the user once used the item. Using similar items could indicate similar interests, user characteristics, and user attributes. When predicting attribute values, we further add additional attribute nodes to represent attribute values, and we create a link between a social node and an attribute node if the user has the attribute value. When predicting attribute categories (i.e., category of majors, category of employers, and country), each additional attribute node represents an attribute category, and we create a link between a social node and an attribute node if the user has some attribute value that belongs to the corresponding attribute category. Table 3 shows the basic statistics of our constructed SBAs for predicting attribute values and attribute categories.

In this article, we set the weights of all links in the SBAs to be 1. Therefore, our attacking result represents a lower bound on what an attacker can achieve in practice. An attacker could leverage machine learning techniques (we discuss one in Section 8) to learn link weights to further improve success rates.

7 EXPERIMENTS

7.1 Experimental Setup

We describe evaluation metrics, training and testing, as well as parameter settings.

Table 3. Basic Statistics of Our SBAs for Predicting Attribute Values and Attribute Categories

#social nodes	#behavior nodes	#attri. nodes	
		attri. val.	attri. cat.
1,111,905	48,706	210	41
#social links	#behavior links	#attri. links	
		attri. val.	attri. cat.
5,328,308	3,635,231	269,997	238,140

Evaluation metrics. All attacks we evaluate essentially assign a score for each candidate attribute value. Given a targeted user v, we predict top-K candidate attribute values that have the highest scores for each attribute including major, employer, and cities lived. We use Precision, Recall, and F-score to evaluate the top-K predictions. In particular, Precision is the fraction of predicted attribute values that belong to v. Recall is the fraction of v's attribute values that are among the predicted K attribute values. We address score ties in the manner described by McSherry and Najork [44]. Precision characterizes how accurate an attacker's inferences are while Recall characterizes how many user attributes are corrected inferred by an attacker. In particular, Precision for top-1 prediction is the fraction of users that the attacker can correctly infer at least one attribute value. F-score is the harmonic mean of Precision and Recall, i.e., we have

$$\text{F-score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

Moreover, we average the three metrics over all targeted users. We compute these metrics similarly to evaluate attribute-category inference. For convenience, we will also use P, R, and F to represent Precision, Recall, and F-Score, respectively.

We also define *performance gain* and *relative performance gain* of one attack $\mathscr{A}$ over another attack $\mathscr{B}$ to compare their relative performances. We take Precision as an example to show our definitions of performance gain and relative performance gain as follows:

$$\textbf{Performance gain:} \quad \Delta\text{P} = \text{Precision}_{\mathscr{A}} - \text{Precision}_{\mathscr{B}}.$$

$$\textbf{Relative performance gain:} \quad \Delta\text{P}\% = \frac{\text{Precision}_{\mathscr{A}} - \text{Precision}_{\mathscr{B}}}{\text{Precision}_{\mathscr{B}}} \times 100\%.$$

Training and testing. In the experiments of inferring attribute values, for each attribute value, we sample five users uniformly at random from the users that have the attribute value and have reviewed at least five items, and we treat them as test (i.e., targeted) users. In total, we have around 1,050 test users. In the experiments of inferring attribute categories, we transform the attribute values of the test users to the corresponding attribute categories. For test users, we remove their attribute links and use them as ground truth. We repeat the experiments 10 times and average the evaluation metrics over the 10 trials.

Paramater settings. In the dividing step, we set equal shares for social neighbors, behavior-sharing social neighbors, and attribute-sharing social neighbors, i.e., $w_S = w_{BS} = w_{AS} = \frac{1}{3}$. The number of iterations to compute the vote capacity vector is $d = \lfloor \log |V_s| \rfloor = 20$, after which the vote capacity vector converges. Unless otherwise stated, we set the backtracking strength $\alpha = 0.1$.

7.2 Compared Attacks

We compare VIAL with friend-based attacks, behavior-based attacks, and attacks that use both social structures and behaviors. These attacks essentially assign a score for each candidate attribute value or attribute category, and return the K attribute values or attribute categories that have the highest scores. Suppose v is a test user and a is an attribute value or an attribute category, and we denote by $S(v, a)$ the score assigned to a for the test user v.

Random. This baseline method computes the fraction of users in the training dataset that have a certain attribute value or attribute category a, and it treats such fraction as the score $S(v, a)$ for all test users.

Friend-based attacks. We compare with three friend-based attacks. They are CN-SAN, AA-SAN, and RWwR-SAN [23]. These attacks were shown to outperform previous attacks such as LINK [23, 64].

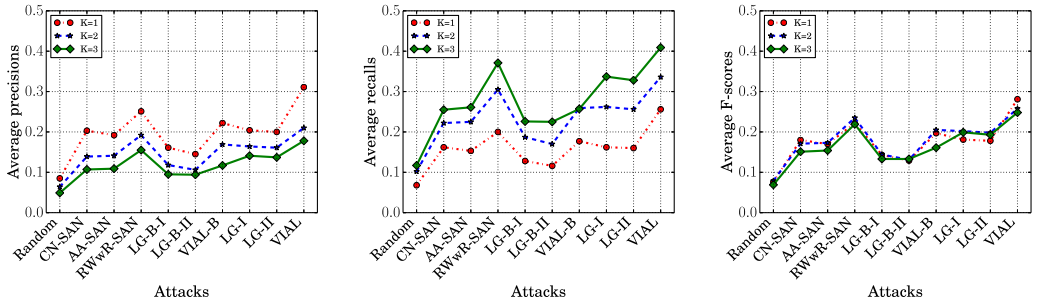
- *CN-SAN.* This attack calculates the number of common social neighbors between v and a as the score $S(v, a)$.
- *AA-SAN.* This attack weights the importance of each common social neighbor between v and a proportional to the inverse of the log of its number of neighbors. Formally, $S(v, a) = \sum_{u \in \Gamma_{v,S} \cap \Gamma_{a,S}} \frac{1}{\log|\Gamma_u|}$.
- *RWwR-SAN.* RWwR-SAN augments the social network with additional attribute nodes. Then it performs a random walk that is initialized from the test user v on the augmented graph. The stationary probability of the attribute node that corresponds to a is treated as the score $S(v, a)$.

Behavior-based attacks. We also evaluate three behavior-based attacks.

- *Logistic regression (LG-B-I)* [60]. LG-B-I treats each attribute value or attribute category as a class and learns a multi-class logistic regression classifier with the training dataset. Specifically, LG-B-I extracts a feature vector whose length is the number of items for each user that has review data, and a feature has a value of the rating score that the user gave to the corresponding item. Google Play allows users to rate or like an item, and we treat a liking as a rating score of 5. For a test user, the learned logistic regression classifier returns a posterior probability distribution over the possible attribute values/categories, which are used as the scores $S(v, a)$. Weinsberg et al. [60] showed that logistic regression classifier outperforms other classifiers including SVM [14] and Naive Bayes [42].
- *Logistic regression with binary features (LG-B-II)* [35]. The difference between LG-B-II and LG-B-I is that LG-B-II extracts binary feature vectors for users. Specifically, a feature has a value of 1 if the user has rated or liked the corresponding item.
- *VIAL-B.* This is a variant of VIAL that only uses behavior data. Specifically, we remove social links from the SBA network and perform our VIAL using the remaining behavior links and attribute links.

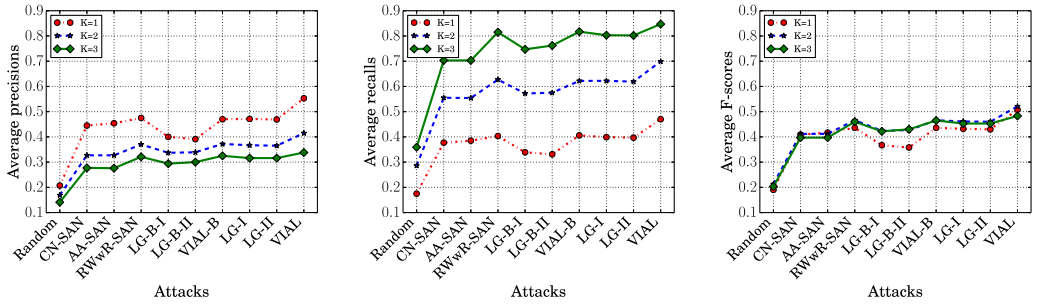
Attacks combining social structures and behaviors. Intuitively, we can combine social structures and behaviors via concatenating social structure features with behavior features. We compare with two such attacks.

- *Logistic regression (LG-I).* LG-I extracts a binary feature vector whose length is the number of users from social structures for each user, and a feature has a value of 1 if the user is a friend of the person that corresponds to the feature. Then LG-I concatenates this feature vector with the one used in LG-B-I and learns multi-class logistic regression classifiers.



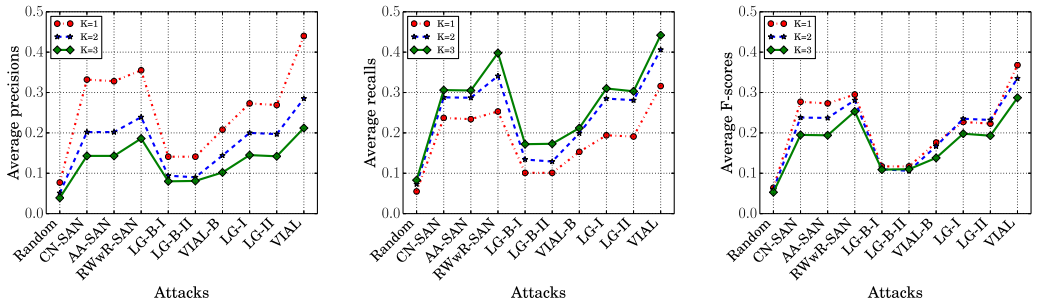
(a) Precision (b) Recall (c) F-score

Fig. 5. Precision, Recall, and F-Score for inferring majors. Although these attacks do not have temporal orderings, we connect them via curves in the figures to better contrast them.



(a) Precision (b) Recall (c) F-score

Fig. 6. Precision, Recall, and F-Score for inferring major categories.



(a) Precision (b) Recall (c) F-score

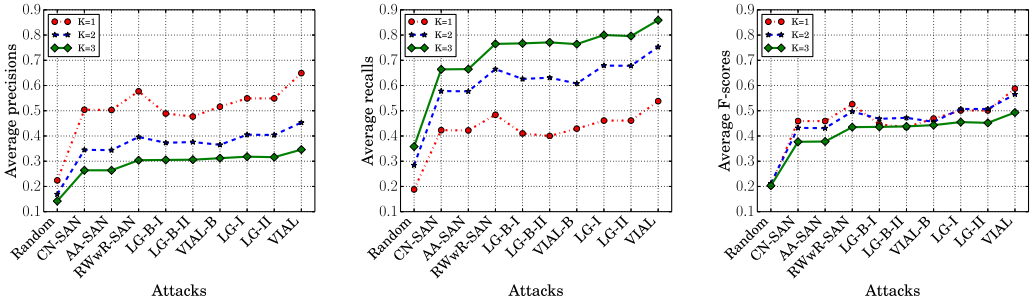
Fig. 7. Precision, Recall, and F-Score for inferring employers.

— *Logistic regression with binary features (LG-II).* LG-II concatenates the binary social structure feature vector with the binary behavior feature vector used by LG-B-II.

We use the popular package LIBLINEAR [17, 39] to learn logistic regression classifiers.

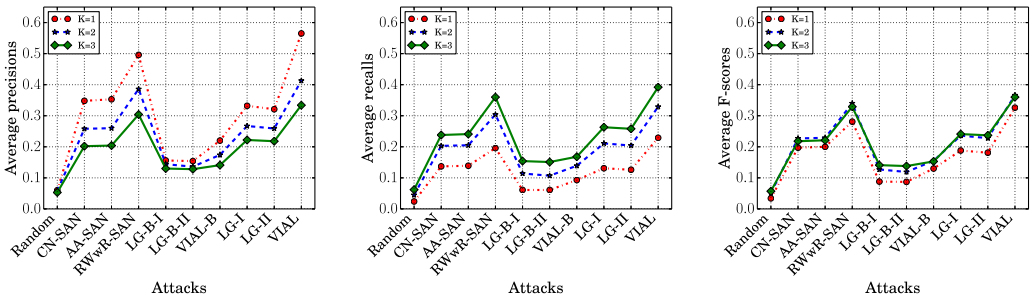
7.3 Results

Figures 5–10 demonstrate the Precision, Recall, and F-score for top-K inference of majors, major categories, employers, employer categories, cities, and countries, where $K = 1, 2, 3$. We average



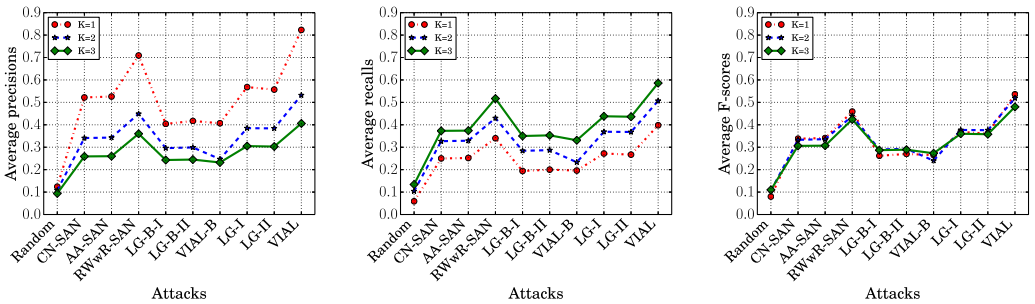
(a) Precision (b) Recall (c) F-score

Fig. 8. Precision, Recall, and F-Score for inferring employer categories.



(a) Precision (b) Recall (c) F-score

Fig. 9. Precision, Recall, and F-Score for inferring cities.



(a) Precision (b) Recall (c) F-score

Fig. 10. Precision, Recall, and F-Score for inferring countries.

the results over 10 trials. We find that standard deviations of the metrics are very small, and thus we do not show them for simplicity. Next, we describe a few key observations we have made from these results.

Comparing friend-based attacks. We find that RWwR-SAN performs the best among the friend-based attacks. Our observation is consistent with the previous work [23]. To better illustrate the difference between the friend-based attacks, we show the performance gains and relative performance gains of RWwR-SAN over other friend-based attacks in Table 4. Please refer to Section 7.1 for formal definitions of (relative) performance gains. The (relative) performance gains are

Table 4. Performance Gains and Relative Performance Gains of RWwR-SAN over Other Friend-based Attacks, Where $K = 1$

(a) Inferring attributes

Attack	ΔP	ΔP%	ΔR	ΔR%	ΔF	ΔF%
CN-SAN	0.07	24%	0.04	24%	0.05	24%
AA-SAN	0.08	26%	0.04	26%	0.05	26%

(b) Inferring attribute categories

Attack	ΔP	ΔP%	ΔR	ΔR%	ΔF	ΔF%
CN-SAN	0.10	19%	0.06	19%	0.07	18%
AA-SAN	0.09	18%	0.06	18%	0.07	17%

We averaged the gains over all attributes or attribute categories. We find that RWwR-SAN is the best friend-based attack.

Table 5. Performance Gains and Relative Performance Gains of VIAL-B over Other Behavior-based Attacks, Where $K = 1$

(a) Inferring attributes

Attack	ΔP	ΔP%	ΔR	ΔR%	ΔF	ΔF%
LG-B-I	0.06	42%	0.04	47%	0.05	45%
LG-B-II	0.07	47%	0.05	52%	0.06	50%

(b) Inferring attribute categories

Attack	ΔP	ΔP%	ΔR	ΔR%	ΔF	ΔF%
LG-B-I	0.03	7%	0.03	8%	0.03	8%
LG-B-II	0.04	8%	0.03	9%	0.04	9%

We averaged the gains over all attributes or attribute categories. We find that VIAL-B is the best behavior-based attack.

averaged over all attributes (i.e., major, employer, and city) or attribute categories (i.e., major category, employer category, and country). The reason why RWwR-SAN outperforms other friend-based attacks is that RWwR-SAN performs a random walk among the augmented graph, which better leverages the graph structure, while other attacks simply count the number of common neighbors or weighted common neighbors.

Comparing behavior-based attacks. We find that VIAL-B performs the best among the behavior-based attacks. Table 5 shows the average performance gains and relative performance gains of VIAL-B over other behavior-based attacks. Our results indicate that our graph-based attack is a better way to leverage behavior structures, compared to LG-B-I and LG-B-II, which flatten the behavior structures into feature vectors. Moreover, LG-B-I and LG-B-II achieve very close performances, which indicates that the rating scores carry little information about user attributes.

Comparing attacks combining social structures and behaviors. We find that VIAL performs the best among the attacks combining social structures and behaviors. Table 6 shows the average performance gains and relative performance gains of VIAL over other attacks. Our results imply that, compared to flattening the structures into feature vectors, our graph-based attack can better integrate social structures and user behaviors.

Table 6. Performance Gains and Relative Performance Gains of VIAL over Other Attacks Combining Social Structures and Behaviors, Where $K = 1$

(a) Inferring attributes

Attack	ΔP	ΔP%	ΔR	ΔR%	ΔF	ΔF%
LG-I	0.17	61%	0.10	65%	0.13	63%
LG-II	0.18	65%	0.11	69%	0.13	67%

(b) Inferring attribute categories

Attack	ΔP	ΔP%	ΔR	ΔR%	ΔF	ΔF%
LG-I	0.15	26%	0.09	26%	0.11	26%
LG-II	0.15	27%	0.09	28%	0.11	27%

We averaged the gains over all attributes or attribute categories. We find that VIAL substantially outperforms other attacks.

Table 7. Performance Gains and Relative Performance Gains of VIAL over Random, RWwR-SAN (The Best Friend-Based Attack), and VIAL-B (The Best Behavior-Based Attack), Where $K = 1$

(a) Inferring attributes

Attack	ΔP	ΔP%	ΔR	ΔR%	ΔF	ΔF%
Random	0.36	526%	0.22	535%	0.27	534%
RWwR-SAN	0.07	20%	0.05	23%	0.06	22%
VIAL-B	0.22	102%	0.13	99%	0.16	100%

(b) Inferring attribute categories

Attack	ΔP	ΔP%	ΔR	ΔR%	ΔF	ΔF%
Random	0.49	306%	0.33	309%	0.39	308%
RWwR-SAN	0.09	14%	0.06	14%	0.07	15%
VIAL-B	0.21	48%	0.12	48%	0.15	48%

We averaged the gains over all attributes or attribute categories.

Comparing VIAL with the best friend-based attack and the best behavior-based attack. Table 7 shows the average performance gains and relative performance gains of VIAL over Random, the best friend-based attack, and the best behavior-based attack. We find that VIAL significantly outperforms these attacks, indicating the importance of combining social structures and behaviors to perform attribute inference. This implies that, when an attacker wants to attack user privacy, the attacker can successfully attack substantially more users using VIAL.

Comparing different attributes. We observe that *inferring attribute categories is easier than inferring attribute values*. For instance, VIAL achieves 0.23 higher F-score for inferring top major category than inferring top major. This is because the number of candidate attribute categories is much smaller than the number of attribute values. Moreover, we find that *cities are easier to infer than employers, which are easier to infer than majors*. For instance, VIAL achieves 0.82 Precision for inferring top country, which is 0.18 higher than inferring top employer category, which in turn achieves 0.09 higher Precision than inferring top major category. One reason might be that some apps are only popular in certain cities [61] and users in different countries use different apps

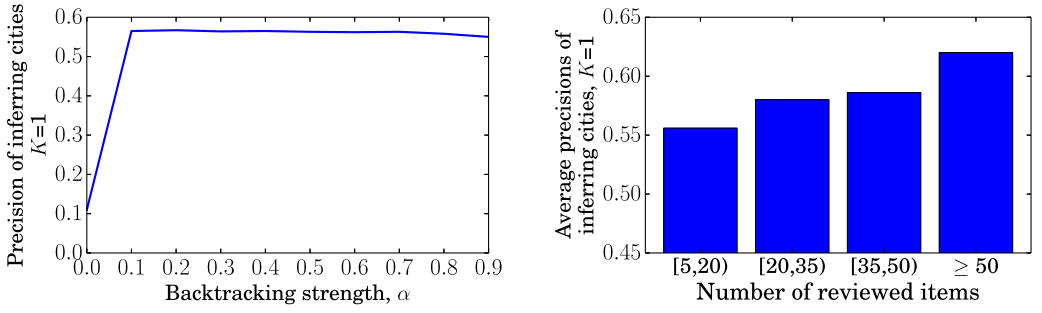


Fig. 11. (a) Impact of the backtracking strength on the Precision of VIAL for inferring cities. We observe that backtracking substantially improves VIAL's performance. (b) Impact of the number of reviewed items on the Precision of our attack VIAL for inferring cities. We observe that, when users share more behaviors, our attack is able to more accurately predict their attributes.

customized to the cultures, e.g., apps that are popular in two countries might use different languages. Such localized apps make it easier to distinguish users from different cities or countries.

Impact of backtracking strength. Figure 11(a) shows the impact of backtracking strength on the Precision of VIAL for inferring cities. According to Theorem 5.1, VIAL with $\alpha = 1$ reduces to random guessing, and thus we do not show the corresponding result in the figure. $\alpha = 0$ corresponds to the case in which VIAL does not use backtracking. We observe that not using backtracking substantially decreases the performance of VIAL. The reason might be that 1) $\alpha = 0$ makes VIAL predict the same attribute values for all test users, according to Theorem 5.2, and 2) a user' attributes are close to the user in the SBA network and backtracking makes it more likely for votes to be distributed among these attribute nodes. Moreover, we find that inference performances are stable across different backtracking strengths once they are larger than 0. We observe similar results for inferring other attributes.

Impact of the number of reviewed items. Figure 11(b) shows the Precision as a function of the number of reviewed items for inferring cities lived. We average Precisions for test users whose number of reviewed items falls under a certain interval (i.e., [5,20), [20,35), [35,50), or $\geq$50). We observe that our attack can more accurately infer attributes for users who share more digital behaviors (i.e., reviewed items in our case).

Confidence estimation. Figure 12 shows the trade-off between the Precision and the fraction of users that are attacked via our confidence estimator. We observe that an attacker can increase the Precision ($K = 1$) of inferring cities from 0.57 to over 0.92 if the attacker attacks half of the test users that are selected via confidence estimation. We also tried the confidence estimator called *gap statistic* [48], in which the confidence score for a targeted user is the difference between the score of the highest ranked attribute value and the score of the second highest ranked one. Our confidence estimator slightly outperforms gap statistic because a test user could have multiple attribute values and our attack could produce close scores for them.

8 DISCUSSION

Difference with conventional Random Walk with Restart (RWwR) [58]: Phase I of VIAL essentially iteratively computes the vote capacity vector according to Equation 7. Therefore, Phase 1 of VIAL can be viewed as performing a *customized* RWwR on the SBA network, where the matrix $\mathbf{M}^T$ and

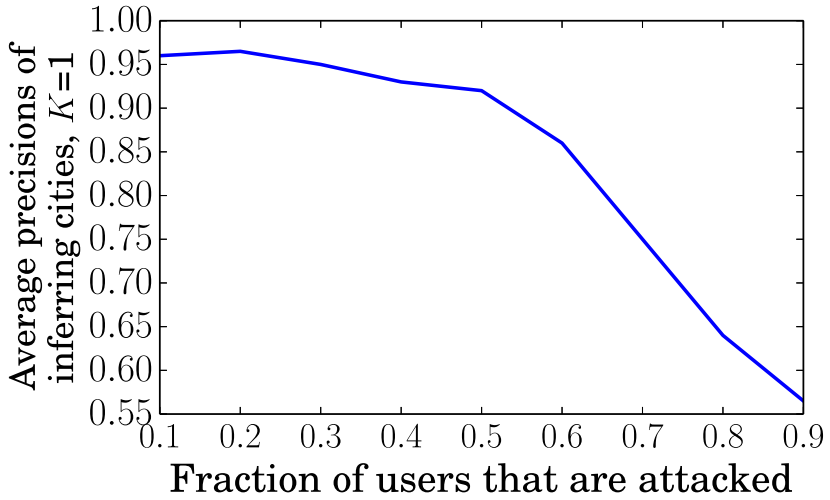


Fig. 12. Confidence estimation: trade-off between the Precision of our attack and the fraction of test users that are attacked. We observe that an attacker can substantially improve the attack success rates when attacking less users.

α are the transition matrix and restart probability of the random walk, respectively. The transition matrix is significantly different from that of the conventional RWwR on an SBA network. Specifically, if we perform a conventional RWwR on an SBA network, the particle in the random walk can stay on any node and move to a node's neighbors or jump back to the starting node in each time step. In our customized RWwR on the SBA network, the particle in the random walk only stays on users (not on behavior nodes nor attribute nodes). Suppose the particle currently stays on a user u. In the next time step, the particle can move to (1) the starting user (i.e., the targeted user) with the restart probability, (2) u's social friends, or (3) other users who share common behaviors or attributes with u.

Learning edge weights. This article focuses on propagating vote capacity among the SBA network with given link weights, and our method VIAL is applicable to any link weights. However, it is an interesting future work to learn the link weights, which could further improve the attacker's success rates. In the following, we discuss one possible approach to learn link weights. Since Phase 1 of VIAL can be viewed as a customized RWwR, the attacker could adapt *supervised random walk* [3] to learn the link weights. Specifically, the attacker already has a set of users with publicly available attributes and the attacker can use them as a training dataset to learn the link weights; the attacker removes these attributes from the SBA network as ground truth, and the link weights are learnt such that VIAL can predict attributes for these users the most accurately.

Directed social networks. In this article, we consider that the social network is undirected. However, directionality of edges may be helpful for attribute inference. Moreover, directed social network and undirected social network may have significantly different structural properties, e.g., their *mixing times* are qualitatively different [47]. It is an interesting future work to generalize our approach to directed social networks.

Limitations. We acknowledge two limitations of our work. First, we only evaluated our attacks using one dataset. To evaluate our attacks, we need datasets that include (1) user attributes, (2) social graph structure, and (3) user behaviors. To the best of our knowledge, there are no publicly

available datasets that satisfy our experimental setup. However, it is an interesting future work to collect other datasets to demonstrate the generalizability of our attacks. Second, we didn't demonstrate the scalability of our attacks. Theoretically, the time complexity of our attack is $O(d \cdot m \cdot n_t)$, where d is the number of iterations needed to compute the vote capacity vector, m is the number of links in the SBA network, and n_t is the number of targeted users. d depends on the structure of the SBA network. It is an interesting future work to study the scalability of our attacks with respect to different SBA networks. Towards this goal, we need to design a model to synthesize SBA networks with different sizes and structures. Existing network generators (e.g., preferential attachment model [4] and SAN model [24]) cannot generate SBA networks. For instance, the preferential attachment model only generates social graphs, while the SAN model generates social graphs with node attributes.

9 RELATED WORK

Attribute inference and *link inference* are two fundamental privacy attacks to social network users. In an attribute inference attack, an attacker aims to infer a target user's missing attributes using publicly available data on social networks. In a link inference attack [23, 38], an attacker aims to infer missing links between users using data publicly available on social networks. Gong et al. [23] proposed a method to jointly perform attribute inference and link inference attacks. Moreover, they demonstrated that attribute inference attacks can be leveraged to improve link inference. In this article, we focus on attribute inference attacks using both social structure and user behavior information.

Friend-based attribute inference. He et al. [27] transformed attribute inference to Bayesian inference on a Bayesian network that is constructed using the social links between users. They evaluated their method using a LiveJournal social network dataset with *synthesized* user attributes. Moreover, it is well known in the machine learning community that Bayesian inference is not scalable. Lindamood et al. [40] modified Naive Bayes classifier to incorporate social links and other attributes of users to infer some attribute. For instance, to infer a user's major, their method used the user's other attributes such as employer and cities lived, the user's social friends and their attributes. However, their approach is not applicable to users that share no attributes at all.

Zheleva and Getoor [64] studied various approaches to consider both social links and groups that users joined to perform attribute inference. They found that, with only social links, the approach LINK achieves the best performance. LINK represents each user as a binary feature vector, and a feature has a value of 1 if the user is a friend of the person that corresponds to the feature. Then LINK learns classifiers for attribute inference using these feature vectors. Gong et al. [23] transformed attribute inference to a link prediction problem. Moreover, they showed that their approaches CN-SAN, AA-SAN, and RWwR-SAN outperform LINK.

Mislove et al. [46] proposed to identify a local community in the social network by taking some seed users that share the same attribute value, and then they predicted all users in the local community to have the shared attribute value. Their approach is not able to infer attributes for users that are not in any local communities. Moreover, this approach is data dependent since detected communities might not correlate with the attribute value. For instance, Trauda et al. [59] found that communities in a MIT male network are correlated with residence but a female network does not have such property. Dong et al. [16] explored the social structures for demographic inference and a factor graph model was proposed to capture the interactions between demographic information and social structures. Chen et al. [12] explored how network characteristics impacted user attribute inference. Chakrabarti et al. [11] proposed a probabilistic method to model relationships between multiple user attributes and their social relationships.

Thomas et al. [57] studied the inference of attributes such as gender, political views, and religious views. They used multi-label classification methods and leveraged features from users' friends and wall posts. Moreover, they proposed the concept of multi-party privacy to defend against attribute inference. Kim and Leskovec [34] proposed a multiplicative attribute graphs (MAG) model to study the interactions between the network structure and the node attributes.

Behavior-based attribute inference. Weinsberg et al. [60] investigated the inference of gender using the rating scores that users gave to different movies. In particular, they constructed a feature vector for each user; the ith entry of the feature vector is the rating score that the user gave to the ith movie if the user reviewed the ith movie, otherwise the ith entry is 0. They compared a few classifiers including logistic regression (LG) [31], SVM [14], and Naive Bayes [42], and they found that LG outperforms the other approaches. Bhagat et al. [7] studied attribute inference in an active learning framework. Specifically, they investigated which movies we should ask users to review in order to improve the inference accuracy the most. However, this approach might not be applicable in real-world scenarios because users might not be interested in reviewing the selected movies.

Chaabane et al. [10] used the information about music users' likes to infer attributes. They augmented the music with the corresponding Wikipedia pages and then used topic modeling techniques to identify the latent similarities between music. A user is predicted to share attributes with those that like similar music with the user. Kosinski et al. [35] tried to infer various attributes based on the list of pages that users liked on Facebook. Similar to the work performed by Weinsberg et al. [60], they constructed a feature vector from the Facebook likes and used LG to train classifiers to distinguish users with different attributes. Luo et al. [41] constructed a model to infer household structures using users' viewing behaviors in Internet Protocol Television (IPTV) systems, and they showed promising results. Nie et al. [49] proposed to infer attributes by combing user behaviors across multiple social medias through a graph-guided model.

Friend-and-behavior-based attribute inference. This article is an extended version of Reference [22]. Different from Reference [22], we performed measurement studies about the Google+ and Google Play dataset, aggregated attributes into attribute categories, and performed attribute inference for attribute categories. Jia et al. [30] recently proposed AttriInfer, a *Markov Random Field* (MRF)-based method to infer attributes in social networks. Specifically, they modeled the social network as an MRF and used Loopy Belief Propagation (LBP) [51] to perform inference. Moreover, since LBP is known to be inefficient for large graphs and not guaranteed to converge for graphs with loops, they optimized LBP to be one order of magnitude more efficient and guaranteed to converge. However, AttriInfer cannot incorporate correlations between attributes, while our VIAL can.

Other approaches. Bonneau et al. [9] studied the extraction of private user data in online social networks via various attacks such as account compromise, malicious applications, and fake accounts. These attacks cannot infer user attributes that users do not provide in their profiles, while our attack can. Otterbacher [50] studied the inference of gender using users' writing styles. Narayanan et al. [48] demonstrated a stronger result, i.e., author identity can be deanonymized via writing style analysis. Zamal et al. [63] used a user's tweets and her neighbors' tweets to infer attributes. They didn't consider social structures nor user behaviors. Gupta et al. [26] tried to infer interests of a Facebook user via sentiment-oriented mining on the Facebook pages that were liked by the user. Golbeck and her collaborators [1, 21] used a user's tweets to infer her personality through participant study. They used both content features and friendship features such as the number of friends and the egocentric network density. However, our attack model is based on the heterogeneous social-behavior-attribute network. Sumneret al. [55] studied dark triad personality

traits prediction through linguistic analysis of tweets. Jurgens [32] used social networks to infer the locations of users. Li et al. [37] proposed a discriminative influence model to profile users' home locations by combining their social networks and the geo-location of the tweets posted by the users. Zhong et al. [65] demonstrated the possibility of inferring user attributes using the list of locations where the user has checked in. These studies are orthogonal to ours since they exploited information sources other than the social structures and behaviors that we focus on.

Attribute inference using social structure and behavior could also be solved by a social recommender system (e.g., Reference [62]). However, such approaches have higher computational complexity than our method for attacking a targeted user, and it is challenging for them to have theoretical guarantees as our attack. For instance, the approach proposed by Ye et al. [62] has a time complexity of $O(m \cdot k \cdot f \cdot d)$ on a single machine, where m is the number of edges, k is the latent topic size, f is the average number of friends, and d is the number of iterations. Note that both our VIAL and this approach can be parallelized on a cluster.

10 CONCLUSION AND FUTURE WORK

In this article, we study the problem of attribute inference via combining social structures and user behaviors that are publicly available in online social networks. To this end, we first propose an SBA network model to gracefully integrate social structures, user behaviors, and their interactions with user attributes. Based on the SBA network model, we design a VIAL to perform attribute inference. We demonstrate the effectiveness of our attack both theoretically and empirically. In particular, via empirical evaluations on a real-world large scale dataset with 1.1 million users, we find that attribute inference is a serious practical privacy attack to online social network users and an attacker can successfully attack more users when considering both social structures and user behaviors. The fundamental reason why our attack succeeds is that private user attributes are statistically correlated with publicly available information, and our attack captures such correlations to map publicly available information to private user attributes.

A few interesting directions for future work include learning the link weights of an SBA network, generalizing VIAL to infer hidden social relationships between users, and defending against our inference attacks.

APPENDIX

A PROOF OF THEOREM 5.1

According to Equation (7), we have

$$\vec{s}_v^{(i)} = (1-\alpha)^i (\mathbf{M}^T)^i \vec{s}_v^{(0)} + \alpha \left(\sum_{k=0}^{i-1} (1-\alpha)^k (\mathbf{M}^T)^k \right) \vec{e}_v. \tag{12}$$

Therefore,

$$\begin{aligned} \lim_{i\to\infty} \vec{s}_v^{(i)} &= \lim_{i\to\infty} \alpha \left(\sum_{k=0}^{i-1} (1-\alpha)^k (\mathbf{M}^T)^k \right) \vec{e}_v \\ &= \alpha (I - (1-\alpha)\mathbf{M}^T)^{-1} \vec{e}_v. \end{aligned} \tag{13}$$

We note that the matrix $(I - (1-\alpha)\mathbf{M}^T)$ is nonsingular because it is strictly diagonally dominant.

B PROOF OF THEOREM 5.2

The matrix $\mathbf{M}$ has non-negative entries, and each row of $\mathbf{M}$ sums to be 1. Therefore, $\mathbf{M}$ can be viewed as a transition matrix. In particular, $\mathbf{M}$ can be viewed as a transition matrix of the following

Markov chain on the SBA network: each social node is a state of the Markov chain; the transition probability from a social node u to another social node x is $\mathbf{M}_{ux}$, i.e., a social node u can only transit to its social neighbors or hop-2 social neighbors with non-zero probabilities.

When the SBA network is connected, the above Markov chain is *irreducible* and *aperiodic*. Therefore, the Markov chain has a unique stationary distribution $\vec{\pi}$. Moreover, according to the Perron-Frobenius theorem [52], we have:

$$\lim_{i\to\infty}(\mathbf{M}^T)^i = [\vec{\pi}\ \vec{\pi}\cdots\vec{\pi}]$$

When $\alpha = 0$, we have $\vec{s}_v^{(i)} = (\mathbf{M}^T)^i\vec{s}_v^{(0)}$. Thus, we have

$$\begin{aligned}\vec{s}_v &= \lim_{i\to\infty}\vec{s}_v^{(i)}\\ &= \lim_{i\to\infty}(\mathbf{M}^T)^i\vec{s}_v^{(0)}\\ &= [\vec{\pi}\ \vec{\pi}\cdots\vec{\pi}]\vec{s}_v^{(0)}\\ &= |V_s|\vec{\pi},\end{aligned}$$

where $|V_s|$ is the sum of the entries of $\vec{s}_v^{(0)}$.

C PROOF OF COROLLARY 5.1

When $w_S = \tau\cdot d_{u,S}$, $w_{BS} = \tau\cdot d_{u,B}$, and $w_{AS} = \tau\cdot d_{u,A}$ for each user u, the Markov chain defined by the transition matrix $\mathbf{M}$ is a random walk on a weighted graph $G_w = (V_w, E_w)$, which is defined as follows: $V_w = V_s$, an edge (u,x) in E_w means that x is u's social neighbor or hop-2 social neighbor in the SBA network, and the weight of the edge $(u,x)\in E_w$ is $\delta_{ux,S}\cdot w_{ux} + \delta_{ux,BS}\cdot d_{u,B}\cdot w_B(u,x) + \delta_{ux,AS}\cdot d_{u,A}\cdot w_A(u,x)$. We can verify that, on the graph G_w, the weights of all edges that are incident to a node u sum to d_u. Therefore, the stationary distribution $\vec{\pi}$ [6] of the random walk on G_w is

$$\vec{\pi} = \left[\frac{d_{u_1}}{D}\ \frac{d_{u_2}}{D}\cdots\frac{d_{u_{|V_s|}}}{D}\right]^T. \tag{14}$$

Thus, according to Theorem 5.2, we have $\vec{s}_{vu} = |V_s|\frac{d_u}{D}$.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous editor and reviewers for their valuable comments and suggestions to improve the quality of the article.

REFERENCES

[1] Sibel Adali and Jennifer Golbeck. 2012. Predicting personality with social behavior. In *Proceedings of the 2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE, 302–309.
[2] Sadia Afroz, Aylin Caliskan-Islam, Ariel Stolerman, Rachel Greenstadt, and Damon McCoy. 2014. Doppelgänger finder: Taking stylometry to the underground. In *IEEE Symposium on Security and Privacy*. San Jose, CA, 212–226.
[3] Lars Backstrom and Jure Leskovec. 2011. Supervised random walks: Predicting and recommending links in social networks. In *WSDM*.
[4] A.-L. Barabási and R. Albert. 1999. Emergence of scaling in random networks. *Science* 286, 5439 (1999), 509–512.
[5] Sergey Bartunov, Anton Korshunov, Seung-Taek Park, Wonho Ryu, and Hyungdong Lee. 2012. Joint link-attribute user identity resolution in online social networks. In *SNA-KDD*.
[6] Ehrhard Behrends. 2000. *Introduction to Markov Chains*. Vieweg.
[7] Smriti Bhagat, Udi Weinsberg, Stratis Ioannidis, and Nina Taft. 2014. Recommending with an agenda: Active learning of private attributes using matrix factorization. In *RecSys*.
[8] Bigfuture major and employer classification. 2014. https://bigfuture.collegeboard.org/majors-careers.

[9] Joseph Bonneau, Jonathan Anderson, and George Danezis. 2009. Prying data out of a social network. In *Proceedings of the 2009 International Conference on Advances in Social Network Analysis and Mining (ASONAM)*.
[10] Abdelberi Chaabane, Gergely Acs, and Mohamed Ali Kaafar. 2012. You are what you like! Information leakage through users' interests. In *Proceedings of the 19th Annual Network & Distributed System Security Symposium*.
[11] Deepayan Chakrabarti, Stanislav Funiak, Jonathan Chang, and Sofus A. Macskassy. 2014. Joint inference of multiple label types in large networks. In *Proceedings of the 31st International Conference on International Conference on Machine Learning—Volume 32 (ICML'14)*. II-874–II-882.
[12] Jiayi Chen, Jianping He, Lin Cai, and Jianping Pan. 2016. Profiling online social network users via relationships and network characteristics. In *Proceedings of the Global Communications Conference (GLOBECOM'16)*. IEEE, 1–6.
[13] Christian Ludl et al. On the effectiveness of techniques to detect phishing sites. *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer Berlin Heidelberg.
[14] Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Mach. Learn.* 20, 273 (1995).
[15] Ratan Dey, Cong Tang, Keith Ross, and Nitesh Saxena. 2012. Estimating age privacy leakage in online social networks. In *INFOCOM*.
[16] Yuxiao Dong, Yang Yang, Jie Tang, Yang Yang, and Nitesh V Chawla. 2014. Inferring user demographics and social strategies in mobile social networks. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 15–24.
[17] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *Journal of Machine Learning Research* 9 (2008), 1871–1874.
[18] Federal Trade Commission. 2014. Data brokers: A call for transparency and accountability. *Federal Trade Commission* (2014).
[19] Oana Goga, Howard Lei, Sree Hari Krishnan Parthasarathi, Gerald Friedland, Robin Sommer, and Renata Teixeira. 2013. Exploiting innocuous activity for correlating users across sites. In *WWW*.
[20] Oana Goga, Daniele Perito, Howard Lei, Renata Teixeira, and Robin Sommer. 2013. *Large-scale Correlation of Accounts Across Social Networks*. Technical report. *International Computer Science Institute*. Technical Report TR-13-002, Berkeley, California.
[21] Jennifer Golbeck, Cristina Robles, and Karen Turner. 2011. Predicting personality with social media. In *CHI'11 Extended Abstracts on Human Factors in Computing Systems (CHI EA'11)*. ACM, 253–262. DOI: http://dx.doi.org/10.1145/1979742.1979614
[22] Neil Zhenqiang Gong and Bin Liu. 2016. You are who you know and how you behave: Attribute inference attacks via users' social friends and behaviors. In *USENIX Security'16*.
[23] Neil Zhenqiang Gong, Ameet Talwalkar, Lester Mackey, Ling Huang, Eui Chul Richard Shin, Emil Stefanov, Elaine Runting Shi, and Dawn Song. 2014. Joint link prediction and attribute inference using a social-attribute network. *ACM Trans. Intell. Syst. Technol.* 5, 2, Article 27 (April 2014).
[24] Neil Zhenqiang Gong, Wenchang Xu, Ling Huang, Prateek Mittal, Emil Stefanov, Vyas Sekar, and Dawn Song. 2012. Evolution of social-attribute networks: Measurements, modeling, and implications using Google+. In *IMC*.
[25] Google Play. 2016. Homepage. Retrieved from https://play.google.com/store?hl=en.
[26] Payas Gupta, Swapna Gottipati, Jing Jiang, and Debin Gao. 2013. Your love is public now: Questioning the use of personal information in authentication. In *AsiaCCS*.
[27] Jianming He, Wesley W. Chu, and Zhenyu Victor Liu. 2006. Inferring privacy information from social networks. In *Proceedings of the International Conference on IEEE Intelligence and Security Informatics*.
[28] Raymond Heatherly, Murat Kantarcioglu, and Bhavani Thuraisingham. 2013. Preventing private information inference attacks on social networks. *IEEE Transactions on Knowledge and Data Engineering* 25, 8 (2013), 1849–1862.
[29] Markus Jakobsson. 2005. Modeling and preventing phishing attacks. In *Financial Cryptography and Data Security. FC 2005*. A. S. Patrick, M. Yung (Eds.). Lecture Notes in Computer Science, vol 3570. Springer, Berlin, Heidelberg.
[30] Jinyuan Jia, Binghui Wang, Le Zhang, and Neil Zhenqiang Gong. 2017. AttriInfer: Inferring user attributes in online social networks using Markov random fields. In *WWW*.
[31] David W. Hosmer Jr and Stanley Lemeshow. 2004. *Applied Logistic Regression*. John Wiley & Sons.
[32] David Jurgens. 2013. That's what friends are for: Inferring location in online social media platforms based on social relationships.*ICWSM* 13 (2013), 273–282.
[33] Tapas Kanungo, D. M. Mount, N.S. Netanyahu, and C. D. Piatko. 2002. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24, 7 (2002), 881–892.
[34] Myunghwan Kim and Jure Leskovec. 2012. Multiplicative attribute graph model of real-world networks. *Internet Math.* 8, 1–2 (2012), 113–160.
[35] Michal Kosinski, David Stillwell, and Thore Graepel. 2013. Private traits and attributes are predictable from digital records of human behavior. *Proceedings of the National Academy of Sciences* 110, 15 (2013), 5802–5805.

[36] Sebastian Labitzke, Florian Werling, and Jens Mittag. 2013. Do online social network friends still threaten my privacy? In *CODASPY*.
[37] Rui Li, Shengjie Wang, Hongbo Deng, Rui Wang, and Kevin Chen-Chuan Chang. 2012. Towards social user profiling: Unified and discriminative influence model for inferring home locations. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'12)*. ACM, 1023–1031.
[38] D. Liben-Nowell and J. Kleinberg. 2003. The link prediction problem for social networks. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management (CIKM'03)*.
[39] LIBLINEAR Package. 2014. http://www.csie.ntu.edu.tw/~cjlin/liblinear/.
[40] Jack Lindamood, Raymond Heatherly, Murat Kantarcioglu, and Bhavani Thuraisingham. 2009. Inferring private information using social network data. In *Proceedings of the 18th International Conference on World Wide Web (WWW'09)*.
[41] Dixin Luo, Hongteng Xu, Hongyuan Zha, Jun Du, Rong Xie, Xiaokang Yang, and Wenjun Zhang. 2014. You are what you watch and when you watch: Inferring household structures from IPTV viewing data. *IEEE Transactions on Broadcasting* 60, 1 (2014), 61–72.
[42] A. McCallum and K. Nigam. 1998. A comparison of event models for naive Bayes text classification. In *AAAI*.
[43] Miller McPherson, Lynn Smith-Lovin, and James M. Cook. 2001. Birds of a feather: Homophily in social networks. *Ann. Rev. Soc.* 27, 415–444.
[44] Frank McSherry and Marc Najork. 2008. Computing information retrieval performance measures efficiently in the presence of tied scores. In *ECIR*.
[45] Tehila Minkus, Yuan Ding, Ratan Dey, and Keith W. Ross. 2015. The city privacy attack: Combining social media and public records for detailed profiles of adults and children. In *COSN*.
[46] Alan Mislove, Bimal Viswanath, Krishna P. Gummadi, and Peter Druschel. 2010. You are who you know: Inferring user profiles in online social networks. In *WSDM*.
[47] Abedelaziz Mohaisen, Huy Tran, Nicholas Hopper, and Yongdae Kim. 2012. On the mixing time of directed social graphs and security implications. In *ASIACCS*.
[48] Arvind Narayanan, Hristo Paskov, Neil Zhenqiang Gong, John Bethencourt, Richard Shin, Emil Stefanov, and Dawn Song. 2012. On the feasibility of internet-scale author identification. In *IEEE S&P*.
[49] Liqiang Nie, Luming Zhang, Meng Wang, Richang Hong, Aleksandr Farseev, and Tat-Seng Chua. 2017. Learning user attributes via mobile social multimedia analytics. *ACM Trans. Intell. Syst. Technol.* 8, 3, Article 36 (April 2017), 19 pages. DOI: http://dx.doi.org/10.1145/2963105
[50] Jahna Otterbacher. 2010. Inferring gender of movie reviewers: Exploiting writing style, content and metadata. In *CIKM*.
[51] J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 2014.
[52] Oskar Perron. 1907. Zur theorie der matrices. *Mathematische Annalen* 64, 2 (1907), 248–263.
[53] Salman Salamatian, Amy Zhang, Flavio du Pin Calmon, Sandilya Bhamidipati, Nadia Fawaz, Branislav Kveton, Pedro Oliveira, and Nina Taft. 2013. How to hide the elephant—or the donkey—in the room: Practical privacy against statistical inference for large data. In *IEEE GlobalSIP*.
[54] Spear Phishing Attacks. 2017. Retrieved fromhttp://www.microsoft.com/protect/yourself/ phishing/spear.mspx.
[55] Chris Sumner, Alison Byers, Rachel Boochever, and Gregory J. Park. 2012. Predicting dark triad personality traits from Twitter usage and a linguistic analysis of Tweets. In *Proceedings of the 2012 11th International Conference on Machine Learning and Applications, Volume 02 (ICMLA'12)*. 386–393.
[56] L. Sweeney. 2002. k-anonymity: A model for protecting privacy. *Int. J. Uncertain., Fuzziness Knowl.-based Syst.* 10, 5 (2002), 557–570.
[57] Kurt Thomas, Chris Grier, and David M. Nicol. 2010. unFriendly: Multi-party privacy risks in social networks. In *PETS*.
[58] Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. 2006. Fast random walk with restart and its applications. In *ICDM*.
[59] Amanda L. Trauda, Peter J. Muchaa, and Mason A. Porter. 2012. Social structure of facebook networks. *Physica A: Statistical Mechanics and its Applications* 391, 16 (2012), 4165–4180.
[60] Udi Weinsberg, Smriti Bhagat, Stratis Ioannidis, and Nina Taft. 2012. BlurMe: Inferring and obfuscating user gender based on ratings. In *RecSys*.
[61] Qiang Xu, Jeffrey Erman, Alexandre Gerber, Zhuoqing Mao, Jeffrey Pang, and Shobha Venkataraman. 2011. Identifying diverse usage behaviors of smartphone apps. In *IMC*.
[62] Mao Ye, Xingjie Liu, and Wang-Chien Lee. 2012. Exploring social influence for recommendation - A probabilistic generative model approach. In *SIGIR*.
[63] Faiyaz Al Zamal, Wendy Liu, and Derek Ruths. 2012. Homophily and latent attribute inference: Inferring latent attributes of Twitter users from neighbors. In *ICWSM*.

[64] E. Zheleva and L. Getoor. 2009. To join or not to join: The illusion of privacy in social networks with mixed public and private user profiles. In *WWW*.
[65] Yuan Zhong, Nicholas Jing Yuan, Wen Zhong, Fuzheng Zhang, and Xing Xie. 2015. You are where you go: Inferring demographic attributes from location check-ins. In *WSDM*.

Received October 2016; revised August 2017; accepted October 2017