AttriInfer: Inferring User Attributes in Online Social Networks Using Markov Random Fields

Jinyuan Jia
Iowa State University
jinyuan@iastate.edu

Binghui Wang
Iowa State University
binghuiw@iastate.edu

Le Zhang
Iowa State University
lezhang@iastate.edu

Neil Zhenqiang Gong
Iowa State University
neilgong@iastate.edu

ABSTRACT

In the attribute inference problem, we aim to infer users' private attributes (e.g., locations, sexual orientation, and interests) using their public data in online social networks. State-of-the-art methods leverage a user's both public friends and public behaviors (e.g., page likes on Facebook, apps that the user reviewed on Google Play) to infer the user's private attributes. However, these methods suffer from two key limitations: 1) suppose we aim to infer a certain attribute for a target user using a training dataset, they only leverage the labeled users who have the attribute, while ignoring the label information of users who do not have the attribute; 2) they are inefficient because they infer attributes for target users one by one. As a result, they have limited accuracies and applicability in real-world social networks.

In this work, we propose *AttriInfer*, a new method to infer user attributes in online social networks. AttriInfer can leverage both friends and behaviors, as well as the label information of training users who have an attribute and who do not have the attribute. Specifically, we model a social network as a *pairwise Markov Random Field (pMRF)*. Given a training dataset, which consists of some users who have a certain attribute and some users who do not have a certain attribute, we compute the *posterior probability* that a target user has the attribute and use the posterior probability to infer attributes. In the basic version of AttriInfer, we use *Loopy Belief Propagation (LBP)* to compute the posterior probability. However, LBP is not scalable to very large-scale real-world social networks and not guaranteed to converge. Therefore, we further optimize LBP to be scalable and guaranteed to converge. We evaluated our method and compare it with state-of-the-art methods using a real-world Google+ dataset with 5.7M users. Our results demonstrate that our method substantially outperforms state-of-the-art methods in terms of both accuracy and efficiency.

Keywords

Attribute inference; Privacy in online social networks; Machine learning as privacy attacks

WWW 2017, April 3–7, 2017, Perth, Australia.
ACM 978-1-4503-4913-0/17/04.
http://dx.doi.org/10.1145/3038912.3052695

1. INTRODUCTION

In an online social network (OSN), a user often has a profile that consists of *a friend list*, *behaviors*, and *attributes* (e.g., cities lived, employer, sexual orientation). On Facebook, behaviors could be the list of pages a user liked/shared; and on Google+, behaviors could be the list of Google Play apps a user liked/reviewed. From a perspective of data science, an OSN is essentially a mixture of both *public data* and *private data*. For instance, public data could include the friend lists, behaviors, and attributes that users make public. Private data could include attributes that users do not disclose in their profiles. We call such attributes *private attributes*. Specifically, private attributes could be 1) attributes that a user sets to be private to its friends using *privacy setting* or 2) attributes that a user does not provide in its profile.

One problem of increasing interest revolves around these private attributes [16, 19, 34, 28, 13, 22, 18, 32, 4, 17, 12]. In this *attribute inference problem*, we first collect public data from an OSN, and then use them to infer private attributes of certain target users via machine learning techniques. Attribute inference has serious implications for Internet privacy as well as applications for targeted advertisements and personalized recommendation. Therefore, various parties (e.g., cyber criminal, online social network provider, advertiser, data broker, and surveillance agency) are motivated to perform attribute inference. For instance, cyber criminals can leverage the inferred user attributes to further perform other attacks, e.g., targeted social engineering attacks and attacking personal information based user authentication (also known as "security questions") [15]; data brokers make profit via selling the inferred user attribute information to other parties such as advertisers, banking companies, and insurance industries [1]. Moreover, an attacker can leverage the inferred attributes to link online users across multiple sites [10, 2] or with offline records (e.g., publicly available voter registration records) [27, 21] to form detailed and composite user profiles, resulting in bigger security and privacy risks.

Conventionally, most attribute inference methods [16, 19, 34, 28, 13, 22, 18, 32, 4, 17] leverage either public social graph or behaviors. Recently, Gong et al. [12] proposed a method (called VIAL) that combines both social graph and behaviors to infer users' private attributes. In a nutshell, VIAL augments a social graph with behavior nodes and attribute nodes; to infer a target user's attributes, VIAL performs a customized *random walk* started from the user among the augmented graph, and the stationary probabilities of attribute nodes are then used to infer attributes of the target user. Gong et al. demonstrated that VIAL achieves state-of-the-art inference accuracy for various attributes. However, VIAL suffers from two major limitations. First, suppose we aim to infer whether a target user has a certain attribute or not, and we are given a training dataset, in which we have some users who have the attribute (*positive* training users) and users who do not (*negative* training users). VIAL cannot leverage the label information of the negative training users because that a user does not have a certain attribute is not encoded in the augmented graph. Second, VIAL is not efficient because it needs to perform the random walk for every target user.

Our work: In this work, we propose AttriInfer, a new method that combines social graph and behaviors to perform attribute inference. AttriInfer can leverage both positive training users and negative training users in the training dataset, as well as is efficient. In particular, AttriInfer can infer attributes for all target users simultaneously. In AttriInfer, we model each user as a random variable that characterizes the user's attribute, and we model the joint probability of all users as a *pairwise Markov Random Field (pMRF)* based on the social network structure. Given a training dataset, we first use behaviors to learn a probability that each user has a considered attribute, and we call such probability *prior probability*. Then, based on the pMRF model, we compute the posterior probability that each target user has the attribute. The posterior probabilities are used to infer attributes. In the basic version of AttriInfer, we use the popular Loopy Belief Propagation (LBP) method to compute the posterior probabilities.

However, the basic version has two shortcomings: 1) we found that it is not scalable enough because LBP needs to maintain messages on each edge, and 2) it is not guaranteed to converge because LBP might oscillate on loopy graphs [25]. Therefore, we further optimize AttriInfer to address these shortcomings. Our optimizations include eliminating message maintenance and approximating AttriInfer by a concise matrix form. We also derive the conditions for our optimized AttriInfer to converge.

We compare AttriInfer with state-of-the-art methods on a large-scale Google+ dataset with 5.7M users. First, we observe that AttriInfer's inference accuracy increases by combining social graph with behaviors. Second, the optimized version of AttriInfer is significantly more efficient than the basic version. Third, AttriInfer substantially outperforms state-of-the-art methods in terms of both inference accuracy and efficiency. For instance, AttriInfer outperforms VIAL by 20% at inferring cities a user lived in.

In summary, our key contributions are as follows:

- We propose AttriInfer, a new attribute inference method based on pairwise Markov Random Field.
- We optimize AttriInfer to be scalable and convergent.
- We compare AttriInfer with state-of-the-art methods using a large-scale dataset. Our results demonstrate that AttriInfer substantially outperforms state-of-the-art methods in terms of both inference accuracy and efficiency.

2. RELATED WORK

Using behaviors: Weinsberg et al. [32] 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), SVM, and Naive Bayes, and found that LG outperforms other approaches. Chaabane et al. [4] used the information about the musics users like to infer attributes. They augmented the musics with the corresponding Wikipedia pages and then used topic modeling techniques to identify the latent similarities between musics. A user is predicted to share attributes with those that like similar musics with the user. Kosinski et al. [17] 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. [32], they constructed a feature vector from the Facebook likes and used LR to train classifiers to distinguish users with different attributes.

Using social graph: Lindamood et al. [19] 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 [34] studied various approaches to consider both social links and groups that users joined to perform attribute inference. Gong et al. [13] transformed attribute inference to a link prediction problem through an augmented graph and demonstrated that their method outperforms various social graph based methods.

Mislove et al. [22] 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. [29] found that communities in a MIT male network are correlated with residence but a female network does not have such property. Thomas et al. [28] 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.

Using behaviors and social graph: Gong et al. [12] proposed to combine behaviors and social graphs to perform attribute inference. In particular, their method (called VIAL) augments the social graph with additional nodes, each of which represents an attribute or a behavior object. If a user has a certain attribute or performed a certain behavior on a behavior object, then VIAL adds an edge between the user and the corresponding attribute node or behavior node. To infer attributes of a target user, VIAL essentially performs a customized random walk started from the target user among the augmented graph, computes the stationary probability distribution of the random walk, and uses it to infer attributes of the target user. VIAL suffers from two limitations. First, VIAL cannot leverage negative training users who do not have a certain attribute, because the augmented graph does not encode the information that a user does not have a certain attribute. Second, VIAL is not scalable because it performs the random walk for every target user.

Other approaches: Bonneau et al. [3] studied the extraction of private user data in OSNs via various attacks such as account compromise, malicious applications, and fake accounts. These attacks can not infer user attributes that users do not provide in their profiles, while our method can. Otterbacher [24] studied the inference of gender using users' writing styles. More recent studies [23, 2] demonstrated stronger results, i.e., authors can be deanonymized via writing style analysis. Zamal et al. [33] used a user's tweets and her neighbors' tweets to infer attributes. They didn't consider social structures nor user behaviors. Gupta et al. [15] tried to infer interests of a Facebook user via sentiment-oriented mining on the pages that were liked by the user. These studies are orthogonal to ours since they exploited information sources other than social graphs and behaviors that we focus on.

Finally, we note that another line of research [5, 8] aims to design new paradigms of OSNs in which users have better control over private data. These studies are orthogonal to ours as we focus on the currently used paradigms of OSNs.

3. PROBLEM FORMULATION

3.1 Categorizing Different Types of Attributes

Binary attributes: A binary attribute only has two possible values, and a user can have either of them. Gender (male vs. female) and political view (democratic vs. republican) are example binary attributes. Note that we distinguish between *attribute* and *attribute value*, e.g., gender is an attribute, while male is an attribute value.

Multi-value attributes: A multi-value attribute has more than two possible attribute values. However, a user only has one value for the attribute. Age (e.g., 0-10 vs. 10-15 vs. 15-20 vs. >20) is an example multi-value attribute.

Multi-value-multi-label attributes: A multi-value-multi-label attribute has more than two possible attribute values, and a user can have more than one attribute value for the attribute. For instance, *cities lived* is a multi-value-multi-label attribute because a person might have lived in multiple cities.

Transforming a multi-value attribute and a multi-value-multi-label attribute to multiple binary attributes: We transform a multi-value attribute or a multi-value-multi-label attribute to multiple binary attributes. Specifically, for each attribute value of the attribute, we create a binary attribute, which has two attribute values "yes" and "no". For instance, for the *cities lived* attribute, we represent each city as a binary attribute. If a user once lived in a certain city, the user's attribute value for the corresponding binary attribute is "yes". We note that among the binary attributes corresponding to a multi-value attribute, a user has "yes" for only one of them. This is because a user only has one value for a multi-value attribute. As we will see later, this transformation makes it easier to model multi-value attributes and multi-value-multi-label attributes.

3.2 Attribute Inference

We take one binary attribute A as an example to illustrate the problem of attribute inference. Suppose we are given an undirected social graph $G = (V, E)$, where a node $v \in V$ represents a user and an edge $(u, v) \in E$ indicates a certain relationship between u and v. For instance, such relationship could be that u and v are friends on Facebook or u and v are in each other' circle on Google+.

Each user either has the attribute A or does not have the attribute A. For instance, when the binary attribute A is a city, having A means that the user lives/lived in the city and not having A means that the user hasn't lived in the city. A user is called *positive user* if it has the attribute, otherwise it is called *negative user*. Moreover, each user has a list of behaviors (though this list might be empty for some users). For instance, a user's behaviors can be the pages liked or shared on Facebook, or the mobile apps liked or reviewed on Google Play. Given these terminologies, we can formally define attribute inference problem as follows:

DEFINITION 1 (ATTRIBUTE INFERENCE PROBLEM). *Suppose we are given 1) a binary attribute A, 2) an undirected social graph $G = (V, E)$, 3) the list of behaviors of each user in the graph, 4) a training dataset consisting of some users who are known to have the attribute A and some users who are known to not have A, and 5) a set of target users. Attribute inference aims to infer whether each target user has A or not.*

3.3 Design Goals

We aim to design a method that achieves the following goals.

1) Leveraging both behaviors and social graph: Previous work [12] has demonstrated that combining behaviors and social graph can achieve better inference accuracies. Therefore, our method should be able to combine the two heterogeneous sources of information.

2) Incorporating training users having and not having the attribute: From users' public data in online social networks, we can often obtain a set of users who publicly disclose that they have the attribute. Moreover, we can also (approximately) obtain a set of users who do not have the attribute. For instance, if the attribute is a city, then we can find the list of users who disclosed that they live/lived in the city and we treat them as positive training users. If a user discloses multiple cities lived but they do not include the considered city, then we can treat the user as a negative training user. The intuition is that such a user would be highly likely to also disclose the considered city in its profile if he/she lives/lived in it.

3) Scalable: Real-world OSNs often have hundreds of millions of users and billions of edges. Moreover, a large fraction of users do not disclose their attributes (e.g., around 70% of Google+ users did not disclose any attributes [14]), and these users are potential target users. In other words, the number of target users is also large. Therefore, our method should be computationally efficient with respect to the size of the OSN as well as the number of target users.

Most existing attribute inference methods [16, 19, 34, 28, 13, 22, 18, 32, 4, 17] do not satisfy requirement 1). VIAL [12] does not satisfy requirements 2) and 3).

3.4 Threat Model

Attribute inference can be viewed as a privacy attack to target users. We discuss the threat model of attribute inference attacks.

Attackers: The attacker could be any party who has interests in user attributes. For instance, the attacker could be OSN provider, advertiser, data broker, or cyber criminal. OSN providers and advertisers could use the user attributes for targeted advertisements; data brokers make profit via selling the user attributes to other parties such as advertisers, banking companies, and insurance industries [1]; cyber criminals can leverage user attributes to perform targeted social engineering attacks (now often referred to as spear phishing attacks) and attacking personal information based user authentication [15].

Attack procedure: In order to use our method to perform attribute inference attacks. An attacker first collects public social graph and user behaviors from a certain OSN. Then, the attacker infers attributes of certain target users using the public data.

Performing further attacks: 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 OSNs. Thus, an attacker could combine user attributes across multiple OSNs to better profile users, and an attacker could leverage the inferred user attributes to do so [2, 10]. Moreover, an attacker can further use the inferred user attributes to link online users with offline records (e.g., voter registration records) [27, 21], 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., cities lived) seem not private for some target users, an attacker could still use them to link users across multiple online sites and with offline records.

4. DESIGN OF ATTRIINFER

4.1 Overview

Suppose we consider a binary attribute A. Given a training dataset, we first use user behaviors to learn a binary classifier for the attribute. Then, we use the classifier to predict the probability that each target user has the attribute A. We call this probability *prior probability*. We use a binary random variable to model each user, and we model the joint probability distribution of all binary random variables as a pairwise Markov Random Field (pMRF) based on the structure of the social network. Given the training dataset and prior probability, we propagate label information among the social graph via the pMRF model. After the propagation, we obtain a posterior probability of having the attribute for each target user. Then, we use the posterior probability to predict whether a target user has the attribute or not. In this section, we introduce a basic version of AttriInfer, in which we use Loopy Belief Propagation (LBP) to infer the posterior probabilities.

4.2 Learning Prior using Behaviors

We associate a binary random variable x_u with each user u, where $x_u = 1$ means that u has the attribute A and $x_u = -1$ means that u does not have the attribute. We denote the behaviors of a user u as a *behavior vector* $\vec{b}_u$. The vector $\vec{b}_u$ can be a binary vector, where an entry is 1 if and only if u has performed a certain action on the corresponding object. For instance, when we consider page likes as behaviors, an entry of 1 means that the user liked the corresponding page. The behavior vector $\vec{b}_u$ can also be a real-valued vector. For instance, when we consider reviews as behaviors, an entry is the rating score that a user gave to the corresponding item (e.g., app, movie, book). Likewise, when we consider clickstream as behaviors, an entry of a user's behavior vector can be the frequency of a certain subsequence (consisting of click events and discretized time gaps between them) that appears in the user's clickstream [30, 31].

We learn the prior probability of target users using a standard logistic regression classifier. Specifically, the probability q_u that u has the attribute A is modeled as $q_u = Pr(x_u = 1) = \frac{1}{1+exp(-h_u)}$, where $h_u = \vec{b}_u^T \cdot \vec{c} + d$. d and the vector $\vec{c}$ are parameters of the logistic regression classifier. Given a training dataset, in which each user has behaviors, we can learn these parameters via *maximum likelihood estimation*. In our experiments, we leverage the library LIBLINEAR [7] to learn these parameters. We note that we choose logistic regression classifier instead of a Support Vector Machine (SVM) because SVM does not directly produce a probability that a user has the attribute.[1] With this logistic regression classifier, we can compute the prior probability for each user who has behaviors. A user has a prior probability of 0.5 if it does not have behaviors.

4.3 Propagating Prior Using Social Graph

We leverage a pMRF to model the social graph, with which we can propagate the prior probability among the social graph to compute a posterior probability that each target user has the attribute.

4.3.1 Intuitions

We denote by Γ_u the set of u's neighbors in the social network, and $\bar{x}_{\Gamma_u}$ the observed states (whether having the attribute or not) of the neighbors. Social networks are known to have the *homophily* property [20], i.e., if we sample an edge (u, v) from a social graph uniformly at random, the two users u and v are highly likely to both have the attribute or both not have the attribute. Based on this homophily intuition, we model the probability that a user u has the attribute A when the user's neighbors' states are already known as follows:

$$Pr(x_u = 1|\bar{x}_{\Gamma_u}) = \frac{1}{1+\exp(-\sum_{v\in\Gamma_u} J_{uv}\bar{x}_v - h_u)} , \qquad (1)$$

[1]We note that SVM's outputs can be transformed into probabilities [7], but they achieve suboptimal performance.

where $J_{uv} > 0$ is the homophily strength between u and v, and h_u characterizes u's prior knowledge obtained through the logistic regression classifier. A higher J_{uv} means that u and v are more likely to have the same state.

4.3.2 A Pairwise Markov Random Field

We find that Equation 1 can be achieved by modeling the social graph as a pairwise Markov Random Field (pMRF). A pMRF defines a joint probability distribution for binary random variables associated with all the users in the social graph. Generally speaking, a pMRF is specified by a *node potential* for each user u, which incorporates prior probability about u, and an *edge potential* for each edge (u, v), which represents correlations between x_u and x_v. In our attribute inference problem, we define a node potential $\phi_u(x_u)$ for user u as

$$\phi_u(x_u) := \begin{cases} q_u & \text{if } x_u = 1 \\ 1 - q_u & \text{if } x_u = -1 \end{cases}$$

and an *edge potential* $\varphi_{uv}(x_u, x_v)$ for the edge (u, v) as

$$\varphi_{uv}(x_u, x_v) := \begin{cases} w_{uv} & \text{if } x_u x_v = 1 \\ 1 - w_{uv} & \text{if } x_u x_v = -1 \end{cases} ,$$

where $w_{uv} := (1 + \exp\{-J_{uv}\})^{-1}$. Then, the following pMRF satisfies Equation 1 for every user.

$$Pr(x_V) = \frac{1}{Z} \prod_{u\in V} \phi_u(x_u) \prod_{(u,v)\in E} \varphi_{uv}(x_u, x_v) ,$$

where $Z = \sum_{x_u} \prod_{u\in V} \phi_u(x_u) \prod_{(u,v)\in E} \varphi_{uv}(x_u, x_v)$ is called the partition function and normalizes the probabilities. $w_{uv} > 0.5$ captures the homophily property. In our definitions, w_{uv} can be interpreted as the probability that two linked users have the same state. In this work, we set $w_{uv} = w > 0.5$ for all edges, and we call *w homophily strength*. However, learning the parameters w_{uv} for different edges would be a valuable future work.

4.3.3 Estimating Posterior Probability using LBP

We compute the *posterior probability distribution* of a user u, i.e., $Pr(x_u) = \sum_{x_{V/u}} Pr(x_V)$. For simplicity, we denote by p_u the posterior probability that u has the attribute, i.e., $p_u = Pr(x_u = 1)$. This posterior probability is used to predict whether a user has the attribute or not. In the basic version of AttriInfer, we use Loopy Belief Propagation (LBP) [25] to estimate the posterior probability distribution $Pr(x_u)$. LBP iteratively passes messages between neighboring users in the graph. Specifically, the message $m_{vu}^{(t)}(x_u)$ sent from v to u in the tth iteration is

$$m_{vu}^{(t)}(x_u) = \sum_{x_v} \phi_v(x_v)\varphi_{vu}(x_v, x_u) \prod_{k\in\Gamma(v)/u} m_{kv}^{(t-1)}(x_v), \qquad (2)$$

where $\Gamma(v)/u$ is the set of all neighbors of v, except the receiver node u. This encodes that each node forwards a product over incoming messages of the last iteration and adapts this message to the respective receiver based on the homophily strength with the receiver. LBP stops when the changes of messages become negligible in two consecutive iterations (e.g., l_1 distance of changes becomes smaller than 10^{-3}) or it reaches the predefined maximum number of iterations T. After LBP halts, we estimate the posterior probability distribution $Pr(x_u)$ as follows:

$$Pr^{(t)}(x_u) \propto \phi_u(x_u) \prod_{k\in\Gamma(u)} m_{ku}^{(t)}(x_u), \qquad (3)$$

which is equivalent to

$$p_u^{(t)} = \frac{q_u \prod\limits_{k\in\Gamma(u)} m_{ku}^{(t)}}{q_u \prod\limits_{k\in\Gamma(u)} m_{ku}^{(t)} + (1-q_u)\prod\limits_{k\in\Gamma(u)}(1-m_{ku}^{(t)})}, \quad (4)$$

where $m_{ku}^{(t)} = m_{ku}^{(t)}(x_u = 1)$ and $1 - m_{ku}^{(t)} = m_{ku}^{(t)}(x_u = -1)$. Note that normalizing $m_{ku}^{(t)}(x_u)$ does not affect the computation of posterior probability distribution of any user. Therefore, for simplicity, we have normalized $m_{ku}^{(t)}(x_u)$ such that $m_{ku}^{(t)}(x_u = 1) + m_{ku}^{(t)}(x_u = -1) = 1$ in the above equation.

We note that pMRF and LBP were also adopted to detect Sybils in OSNs [11]. Sybil detection and attribute inference for binary attributes are algorithmically similar. However, previous work [11] didn't perform optimization and convergence analysis as we will discuss in the next two sections Section 5 and Section 6.1.

5. OPTIMIZING ATTRIINFER

The basic version of AttriInfer has two shortcomings: 1) AttriInfer is not scalable enough, and 2) AttriInfer is an iterative method but is not guaranteed to converge. Being not convergent makes it hard to select an appropriate number of iterations. The reason is that LBP maintains messages on each edge and LBP might oscillate on loopy graphs [25]. Therefore, we further optimize AttriInfer to address these shortcomings.

5.1 Eliminating Message Maintenance

One of the major reasons why basic AttriInfer is not scalable enough is that LBP maintains messages on each edge. We observe that the key reason why LBP needs to maintain messages on edges is that when a node v prepares a message to its neighbor u, it excludes the message that u sent to v. Therefore, our first optimization step is to include the message that u sent to v when v prepares its message for u. Formally, we approximate Equation 2 as:

$$m_{vu}^{(t)}(x_u) = \sum_{x_v} \phi_v(x_v)\varphi_{vu}(x_v, x_u) \prod_{k\in\Gamma(v)} m_{kv}^{(t-1)}(x_v). \quad (5)$$

Considering Equation 3, we have:

$$m_{vu}^{(t)}(x_u) \propto \sum_{x_v} \varphi_{vu}(x_v, x_u) Pr^{(t-1)}(x_v). \quad (6)$$

Recall that we normalize $m_{vu}^{(t)}(x_u)$ such that $m_{vu}^{(t)}(x_u = 1) + m_{vu}^{(t)}(x_u = -1) = 1$, and we abbreviate $m_{vu}^{(t)}(x_u = 1)$ as $m_{vu}^{(t)}$. With such normalization, our new messages become:

$$m_{vu}^{(t)} = p_v^{(t-1)} w + (1 - p_v^{(t-1)})(1 - w). \quad (7)$$

AttriInfer does not need to maintain messages on edges using our new messages.

5.2 Linearizing as a Matrix Form

AttriInfer iteratively applies Equations 7 and 4 using our new messages, which still cannot guarantee convergence. The key reason is that Equation 4 combines messages from a user's neighbors *nonlinearly*. We make AttriInfer converge via linearizing Equation 4. The resulting optimized AttriInfer can be represented in a concise matrix form. Before introducing our linearization, we define some terms. In particular, we define the *residual* $\hat{y}$ of a variable y as $\hat{y} = y - 0.5$; and we define the *residual vector* $\hat{\mathbf{y}}$ of $\mathbf{y}$ as $\hat{\mathbf{y}} = [y_1 - 0.5, y_2 - 0.5, \cdots]$.

For convenience, we denote by $\mathbf{p}^{(t)} = [p_1^{(t)}; p_2^{(t)}; \cdots; p_{|V|}^{(t)}]$ a column vector all users' posterior probability in the tth iteration, and its residual vector as $\hat{\mathbf{p}}^{(t)}$. Similarly, we denote by a column vector $\mathbf{q} = [q_1; q_2; \cdots; q_{|V|}]$ all users' prior probability, and by $\hat{\mathbf{q}}$ its residual vector. Moreover, we denote by $\mathbf{M} \in \mathbb{R}^{|V|\times|V|}$ the adjacency matrix of the social graph. With these notations, we can approximate AttriInfer as a concise matrix form. Formally, we have the following theorem.

THEOREM 1. *We can approximate Equations. 7 and 4 as the following equation:*

$$\hat{\mathbf{p}}^{(t)} = \hat{\mathbf{q}} + 2\cdot\hat{w}\cdot\mathbf{M}\cdot\hat{\mathbf{p}}^{(t-1)}. \quad (8)$$

PROOF. See Appendix A. □

Theorem 1 enables us to design an efficient version of AttriInfer: first, we learn the prior probability for each user using behaviors; second, we iteratively apply Equation 8 to compute the posterior probabilities for all target users simultaneously. AttriInfer halts when the relative change of the posterior probability vector in two consecutive iterations is smaller than 10^{-3}.

We note that Gatterbauer et al. [9] also linearized LBP over a pMRF, and their linearized version is similar to ours. However, their linearization requires different heuristics and assumptions. For instance, their linearization requires residual of the homophily strength to be very small and the approximation of $\frac{\frac{1}{k}+\varepsilon_1}{1+\varepsilon_2} \approx \frac{1}{k} + \varepsilon_1 - \frac{\varepsilon_2}{k}$ when ε_1 and ε_2 are very small. In contrast, our linearization does not require these assumptions/approximations. Moreover, we eliminate message maintenance before linearizing Equation 4, which makes linearization much simpler, while their approach does not.

6. THEORETICAL ANALYSIS

6.1 Convergence Analysis

We analyze the condition when the optimized version of AttriInfer converges. Such analysis guides us to select appropriate homophily strength w. Suppose we are given an iterative linear process: $\mathbf{y}^{(t)} \leftarrow \mathbf{c} + \mathbf{D}\mathbf{y}^{(t-1)}$. According to [26], the linear process converges with any initial choice $\mathbf{y}^{(0)}$ if and only if the *spectral radius* of the matrix $\mathbf{D}$ is smaller than 1, i.e., $\rho(\mathbf{D}) < 1$. The spectral radius of a square matrix is the maximum of the absolute values of its eigenvalues. Given this general result, we are able to analyze the convergence condition of AttriInfer.

THEOREM 2 (SUFFICIENT AND NECESSARY CONDITION). *The sufficient and necessary condition that AttriInfer converges is to bound the residual homophily strength $\hat{w}$ as:*

$$\hat{w} < \frac{1}{2\rho(\mathbf{M})}. \quad (9)$$

PROOF. Let $\mathbf{D} = 2\hat{w}\cdot\mathbf{M}$, then $\rho(\mathbf{D}) = 2\hat{w}\cdot\rho(\mathbf{M})$. $\rho(\mathbf{D}) < 1$ holds if and only if $\hat{w} < \frac{1}{2\rho(\mathbf{M})}$. □

Theorem 2 provides a strong sufficient and necessary convergence condition. However, in practice setting $\hat{w}$ using Theorem 2 is computationally expensive, as it involves computing the largest eigenvalue with respect to spectral radius. Hence, we instead derive a *sufficient condition* for AttriInfer's convergence, which enables us to set $\hat{w}$ with cheap computation. Specifically, our sufficient condition is based on the fact that any norm is an upper bound of the spectral radius [6], i.e., $\rho(\mathbf{D}) \le \|\mathbf{D}\|$, where $\|\cdot\|$ indicates some matrix norm. In particular, we use the induced l_1 matrix norm $\|\cdot\|_1$, where $\|\mathbf{D}\|_1 = \max_j \sum_i |\mathbf{D}_{ij}|$, the maximum absolute column sum of the matrix. In this way, our sufficient condition for convergence is as follows:

THEOREM 3 (SUFFICIENT CONDITION). *The sufficient condition that AttriInfer converges is*

$$\hat{w} < \frac{1}{2\|\mathbf{M}\|_1} = \frac{1}{2\max_{u\in V} d_u}, \tag{10}$$

where $d_u = |\Gamma_u|$ is the degree of u.

PROOF. As $\rho(\mathbf{D}) \leq \|\mathbf{D}\|_1$, we achieve the sufficient condition by enforcing $\|\mathbf{D}\|_1 < 1$, where $\mathbf{D} = 2\hat{w}\cdot\mathbf{M}$. Specifically, we have

$$2\hat{w}\cdot\rho(\mathbf{M}) < 1 \Longleftarrow 2\hat{w}\|\mathbf{M}\|_1 < 1 \tag{11}$$

$$\Longleftarrow \hat{w} < \frac{1}{2\|\mathbf{M}\|_1} = \frac{1}{2\max_{u\in V} d_u}. \tag{12}$$

□

Theorem 3 guides us to set $\hat{w}$, i.e., once $\hat{w}$ is smaller than the inverse of 2 times of the maximum node degree, AttriInfer is guaranteed to converge. In practice, however, some users (e.g., celebrities) could have orders of magnitude bigger degrees than the others (e.g., ordinary people), and such nodes make $\hat{w}$ very small. In our experiments, we find that AttriInfer can still converge when replacing the maximum node degree with the average node degree.

6.2 Complexity Analysis

We use sparse matrix representation to implement AttriInfer. Computational time of the optimized AttriInfer consists of two parts: time required to compute the prior probabilities using logistic regression, and time required to compute the posterior probabilities via iterative computations. The time complexity of the second part is $O(t\cdot|E|)$, where t is the number of iterations. The time complexity of AttriInfer does not rely on the number of target users whose attributes an attacker wants to infer. We note that the basic version of AttriInfer has the same asymptotic time complexity. However, the constants in the asymptotic representations are different, which results in different scalability.

VIAL [12] does not learn the prior probabilities. For each target user, VIAL computes the stationary probability distribution of a random walk in the augmented graph, which starts from the target user. The time complexity of VIAL is $O(t\cdot(|E|+m_1+m_2)\cdot n_t)$, where t is the number of iterations that a random walk requires to converge, m_1 is the total number of public attributes of all users, m_2 is the total number of behaviors of all users, and n_t is the number of target users. As we will demonstrate in our experiments, although VIAL does not learn the prior probabilities, it will be orders of magnitude slower than AttriInfer when an attacker aims to infer attributes for a large number of target users.

7. EVALUATIONS

7.1 Experimental Setups

Dataset description: We obtained a Google+ social network snapshot with public user attributes from [14]. And we collected the list of items (e.g., apps, books, musics) that each user reviewed on Google Play, using the same methodology in [12]. A user reviewed an item if the user liked or rated the item. When a user rated an item, the user gave a rating score (1,2,3,4,or 5) to represent its preference. We treat review as behavior. Table 1 summarizes the basic statistics of the dataset. We note that our dataset is much larger than the one used by VIAL (VIAL was tested on a dataset with 1.1 million users). However, we acknowledge that the Google+ dataset might not be a representative sample of the entire Google+ social network, and thus our results might not be representative.

Table 1: Dataset statistics.

#Users	#Edges	#Behaviors
5,735,175	30,644,909	35,528,322

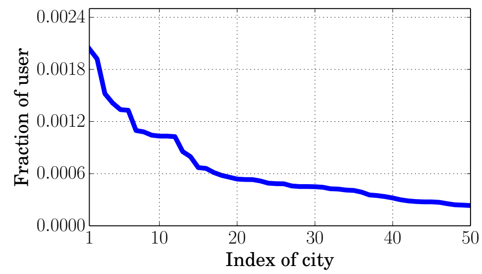


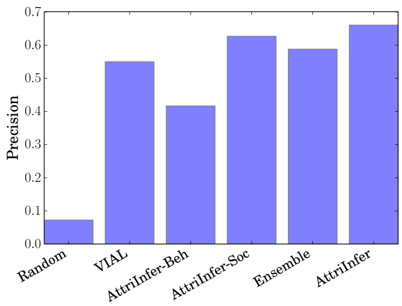
Figure 1: Fraction of users who claim a city as *cities lived*.

We perform evaluations using the attribute *cities lived*. We select top-50 cities in the dataset. In other words, the cities lived attribute has 50 possible attribute values. We note that, our method is also applicable to other attributes such as major, employer, and sexual orientation. We select cities lived because this attribute is available and allows us to perform evaluation at a large scale. 3.25% of users have disclosed at least one of these cities as their cities lived. Figure 1 shows the fraction of users in each city. Cities lived is a multi-value-multi-label attribute, because a user could have lived in multiple cities. We transform the attribute into binary attributes. Since we focus on top-50 cities, we represent cities lived as 50 binary attributes, each of which represents one city.

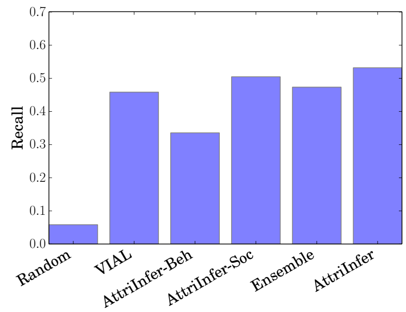
Training and testing: We select 25% of users (i.e., around 7000 users) who have at least one city and 10 behaviors as *test/target users*, and the remaining users who have at least one of the considered cities are treated as *training users*. For a city, the training users who have this city are positive training instances, while the training users who do not have this city are treated as negative training instances. In our method, we train a logistic regression classifier for each city using users' reviews in order to learn prior probabilities. Specifically, a user's behavior vector is a real-valued vector, where an entry is the rating score that the user gave to the corresponding item. If the user only liked an item, the corresponding entry has a value of 5, i.e., we treat a like as a rating with a score of 5. We used the LIBLINEAR library [7] to train linear logistic regression classifiers. We remove the cities of the test users as groundtruth, use an attribute inference method to predict cities for them, and evaluate the performance of the method using the groundtruth.

Compared methods: We will evaluate the following methods. For each test user, every method essentially computes a score for every considered city. For instance, in our method AttriInfer, the score is the posterior probability that the test user lived in the city. In other words, each method computes 50 scores for each test user.

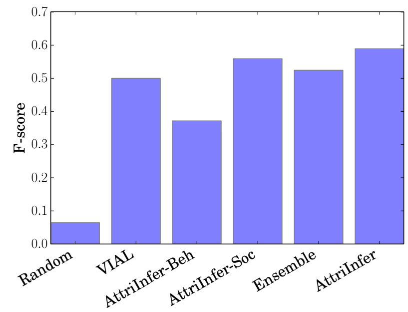
- **Random.** This method computes the fraction of training users who have a city, and predicts this fraction as the score for the city for every test user.
- **VIAL [12].** VIAL combines both social graph and behaviors through an augmented graph. VIAL needs to repeat for every test user. We use the same parameters for VIAL as in [12].
- **AttriInfer.** Our proposed method, which combines social graph and behaviors, as well as computes the scores for all test users



(a) Precision



(b) Recall



(c) F-Score

Figure 2: Precision, Recall, and F-Score of compared methods.

simultaneously. We set the residual of homophily strength (i.e., $\hat{w}$) as $1/(2 \times \textit{average degree})$ of the social graph, to consider convergence. By default, we will use the optimized version.

- **AttriInfer-Beh.** A variant of our method, which only uses behaviors. Specifically, AttriInfer-Beh assigns the prior probability (learnt using behaviors and logistic regression) that a test user has a city as the score for the city.
- **AttriInfer-Soc.** Another variant of our method, which only uses social graph. Specifically, we do not learn prior probability using behaviors. Instead, we assign a prior probability of 0.5 for every user that is not in the training dataset. Moreover, when inferring a city, we assign a prior probability of 0.9 to a positive training user who has the city and 0.1 to a negative training user who does not have the city.
- **Ensemble.** One natural way to combine behaviors and social graph is to use an ensemble method. Therefore, we also evaluate an ensemble method that combines the results of AttriInfer-Beh and AttriInfer-Soc. Specifically, for a test user and a city, each of the two methods produces a score, and we select the larger one as the final score.

Evaluation metrics: We predict top-K cities with the highest scores for a test user, and evaluate the predictions using *Precision*, *Recall*, and *F-Score*. Precision is the fraction of predicted cities that truly belong to the test users, Recall is the fraction of test users' true cities that are among the predictions, and F-Score is the harmonic mean of Precision and Recall, i.e., F-Score $= \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision}+\text{Recall}}$.

7.2 Results

Our method is more accurate than state-of-the-art methods: Figure 2 shows the Precision, Recall, and F-Score of all compared methods for top-1 predictions. We make several observations. First, AttriInfer substantially outperforms VIAL and Ensemble that combine behaviors and social graph. For instance, AttriInfer improves upon VIAL by 20.1%, 16.04%, and 17.86% (these are relative improvements) for Precision, Recall, and F-Score, respectively. The reason is that AttriInfer leverages both positive training users and negative training users, while VIAL only leverages positive training users. Ensemble method's performance is even worse than AttriInfer-Soc, which only uses social graph. This observation demonstrates that AttriInfer is a significantly better way to combine behaviors and social graph.

Second, combining behaviors and social graph indeed improves prediction accuracy. Specifically, AttriInfer outperforms AttriInfer-Beh and AttriInfer-Soc by 58.5% and 5.38%. The reason is that behaviors and social graph are complementary information sources for some users, and combining them can better characterize users.

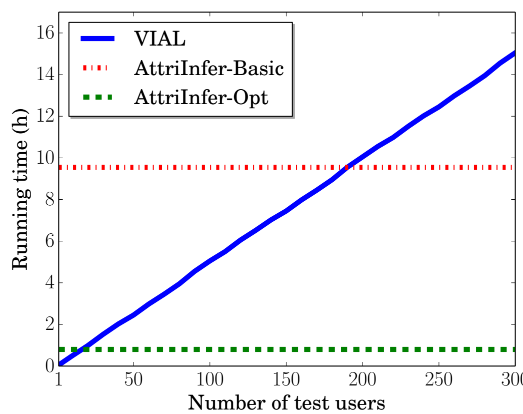


Figure 3: Computational times of VIAL, basic version of AttriInfer (AttriInfer-Basic), and optimized version of AttriInfer (AttriInfer-Opt) as we attack more users.

Our method is more efficient than state-of-the-art methods: Figure 3 shows the times required by VIAL, basic version of AttriInfer (denoted as AttriInfer-Basic), and optimized version of AttriInfer (denoted as AttriInfer-Opt) as we attack more users. In order to better contrast the crossing points of the three curves, we only show the number of test users upto 300 in Figure 3. First, computational times of AttriInfer-Basic and AttriInfer-Opt do not depend on the number of test/target users, because they compute the posterior probabilities for all target users simultaneously. Therefore, their computational times are horizontal lines in the figure. Note that the computational times also include the times required to learn the prior probabilities using behaviors. Second, AttriInfer-Opt is substantially more efficient than AttriInfer-Basic. This is because AttriInfer-Opt does not maintain messages on each edge and is concisely represented as a matrix form, while AttriInfer-Basic maintains messages on each edge. Third, the computational time of VIAL increases linearly as we attack more users, which is consistent with theoretical analysis in [12]. Fourth, when the number of target users is larger than 14, AttriInfer-Opt is more efficient than VIAL, while AttriInfer-Basic is more efficient than VIAL when attacking more than 191 users. Moreover, as we attack more users, the advantage of AttriInfer over VIAL is more significant. AttriInfer iteratively computes the posterior probability vector for each city, while VIAL computes the scores for all considered cities for a test user using one random walk. Therefore, AttriInfer is slower than VIAL when attacking a very small number of users.

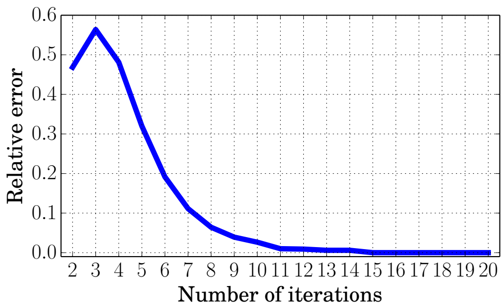


Figure 4: Relative errors of the posterior probability vectors of AttriInfer-Opt in two consecutive iterations as a function of the number of iterations.

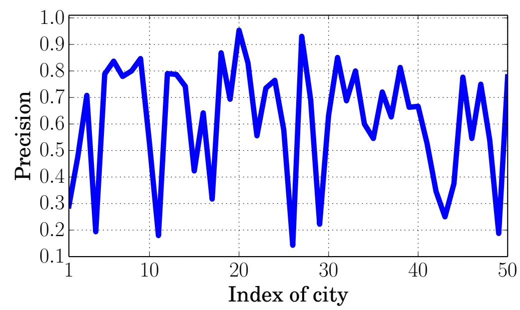


Figure 5: Precision for top-1 prediction of each city. Cities are sorted in a descending order using their populations.

AttriInfer is convergent: Figure 4 demonstrates that the optimized version of AttriInfer indeed converges. Specifically, we define a relative error of the posterior probability vector in the tth iteration as $\|\mathbf{p}^{(t)} - \mathbf{p}^{(t-1)}\|_1 / \|\mathbf{p}^{(t)}\|_1$, and Figure 4 shows the relative error as a function of the number of iterations. As we can see, AttriInfer converges after around 10 iterations.

Prediction accuracy of a city is not significantly correlated with its population: One natural question is which cities are easier to predict. Since we model each city as a binary attribute and AttriInfer produces a posterior probability for each test user, we can measure performance for each city individually. First, we find that the Pearson's correlation coefficient between cities' Precision for top-1 predictions and cities' population is around -0.06, where a city's population is the number of users who claim to live in the city. This implies that cities' prediction Precision is very weakly negatively correlated with their populations. Figure 5 further shows the prediction Precision for each considered city, where the cities are sorted in a descending order using their populations (i.e., the ordering of cities is the same as that in Figure 1). The cities' prediction Precision fluctuates with respect to their populations.

Second, we speculate that popular international cities are harder to predict, because a more diverse set of people may live or have lived in such cities. Indeed, we find that 9 of the 10 cities with the lowest Precision (ranging from 0.14 to 0.38) are popular international cities in US, including San Francisco, Denver, Los Angeles, San Diego, Seattle, New York, Philadelphia, Orlando, and Dallas. People living in these cities are from different cultures, form different friend communities, and use quite different apps. The top-3 cities with the highest Precision (ranging from 0.95 to 0.86) are Istanbul (in Turkey), Bangkok (in Thailand), and Moscow (in Russian), which are less international cities.

8. CONCLUSION AND FUTURE WORK

In this work, we propose AttriInfer, a new method to infer private user attributes in online social networks. AttriInfer can combine both behaviors and social graph, leverage both positive training users and negative training users, and is scalable to large-scale online social networks. Specifically, in AttriInfer, we associate a binary random variable with each user; the binary random variable characterizes whether a user has a considered attribute or not. AttriInfer first learns a prior probability that a user has the attribute using users' behaviors through a logistic regression classifier. Then, AttriInfer models the joint probability distribution of all binary random variables as a pairwise Markov Random Field (pMRF), and computes the posterior probability that each target user has the attribute. In the basic version, AttriInfer uses Loopy Belief Propagation (LBP) to estimate the posterior probabilities under the pMRF. However, the basic version is not scalable enough and is not guaranteed to converge. Therefore, we further optimize AttriInfer to be scalable and guaranteed to converge. We compare AttriInfer with state-of-the-art methods using a large-scale Google+ dataset with 5.7M users. Our results demonstrate that AttriInfer substantially outperforms state-of-the-art methods in terms of both inference accuracy and scalability. Moreover, the optimized version of AttriInfer is substantially more scalable than the basic version. Interesting directions for future work include leveraging novel clickstream features [30, 31] to learn prior probabilities and defending against attribute inference attacks.

APPENDIX

A. PROOF OF THEOREM 1

Our core idea is to linearize Equation 4. We denote $\mathscr{Z}_u^{(t)} = q_u \prod_{v\in\Gamma(u)} m_{vu}^{(t)} + (1-q_u)\prod_{v\in\Gamma(u)}(1-m_{vu}^{(t)})$. By rewriting $p_u^{(t)} = \frac{1}{\mathscr{Z}_u^{(t)}} q_u \prod_{v\in\Gamma(u)} m_{vu}^{(t)}$ with the corresponding residual variables, we have:

$$
\begin{aligned}
& 0.5+\hat{p}_u^{(t)} = \frac{1}{\mathscr{Z}_u^{(t)}}(0.5+\hat{q}_u)\prod_{v\in\Gamma(u)}\left(0.5+\hat{m}_{vu}^{(t)}\right) \\
& \Longrightarrow \ln(1+2\hat{p}_u^{(t)}) = -\ln\mathscr{Z}_u^{(t)} + \ln(1+2\hat{q}_u) + \sum_{v\in\Gamma(u)} \ln\left(0.5+\hat{m}_{vu}^{(t)}\right) \\
& = -\ln\mathscr{Z}_u^{(t)} + \ln(1+2\hat{q}_u) + \sum_{v\in\Gamma(u)} \ln\left(0.5\right) + \sum_{v\in\Gamma(u)} \ln(1+2\hat{m}_{vu}^{(t)})
\end{aligned}
$$

Using approximation $\ln(1+y)\approx y$ when y is small, we have:

$$
2\hat{p}_u^{(t)} = -\ln\mathscr{Z}_u^{(t)} + 2\hat{q}_u + |\Gamma(u)|\cdot\ln(0.5) + \sum_{v\in\Gamma(u)} 2\hat{m}_{vu}^{(t)}. \quad (13)
$$

Similarly, via rewriting $1-p_u^{(t)} = \frac{1}{\mathscr{Z}_u^{(t)}}(1-q_u)\prod_{v\in\Gamma(u)}(1-m_{vu}^{(t)})$ with the corresponding residual variables and using approximation $\ln(1-y)\approx -y$ when y is small, we have:

$$
-2\hat{p}_u^{(t)} = -\ln\mathscr{Z}_u^{(t)} - 2\hat{q}_u + |\Gamma(u)|\cdot\ln(0.5) - \sum_{v\in\Gamma(u)} 2\hat{m}_{vu}^{(t)}. \quad (14)
$$

Adding Equation 13 with Equation 14 yields $\ln\mathscr{Z}_u^{(t)} = |\Gamma(u)|\cdot\ln(0.5)$. Via substituting this relation into Equation 13 or Equation 14, we have $\hat{p}_u^{(t)} = \hat{q}_u + \sum_{v\in\Gamma(u)}\hat{m}_{vu}^{(t)}$. Moreover, by substituting variables in Equation 7 with their residuals, we can represent the residual message $\hat{m}_{vu}^{(t)}$ as $\hat{m}_{vu}^{(t)} = 2\hat{p}_v^{(t-1)}\hat{w}$. Therefore, we have $\hat{p}_u^{(t)} = \hat{q}_u + 2\cdot\hat{w}\cdot\sum_{v\in\Gamma(u)}\hat{p}_v^{(t-1)}$. Using matrix notations, we have Theorem 1.

B. REFERENCES

[1] Data brokers: a call for transparency and accountability. *Federal Trade Commission* (2014).

[2] AFROZ, S., CALISKAN-ISLAM, A., STOLERMAN, A., GREENSTADT, R., AND MCCOY, D. Doppelgänger finder: Taking stylometry to the underground. In *IEEE S & P* (2014).

[3] BONNEAU, J., ANDERSON, J., AND DANEZIS, G. Prying data out of a social network. In *ASONAM* (2009).

[4] CHAABANE, A., ACS, G., AND KAAFAR, M. A. You are what you like! information leakage through users' interests. In *NDSS* (2012).

[5] CRISTOFARO, E. D., SORIENTE, C., TSUDIK, G., AND WILLIAMS, A. Hummingbird: Privacy at the time of twitter. In *IEEE S & P* (2011).

[6] DERZKO, N., AND PFEFFER, A. Bounds for the spectral radius of a matrix. *Mathematics of Computation 19*, 89 (1965).

[7] FAN, R.-E., CHANG, K.-W., HSIEH, C.-J., WANG, X.-R., AND LIN, C.-J. Liblinear: A library for large linear classification. *JMLR 9* (2008), 1871–1874.

[8] FELDMAN, A. J., BLANKSTEIN, A., FREEDMAN, M. J., AND FELTEN, E. W. Social networking with frientegrity: Privacy and integrity with an untrusted provider. In *USENIX Security Symposium* (2012).

[9] GATTERBAUER, W., GÜNNEMANN, S., KOUTRA, D., AND FALOUTSOS, C. Linearized and single-pass belief propagation. *PVLDB* (2015).

[10] GOGA, O., LEI, H., PARTHASARATHI, S. H. K., FRIEDLAND, G., SOMMER, R., AND TEIXEIRA, R. Exploiting innocuous activity for correlating users across sites. In *WWW* (2013).

[11] GONG, N. Z., FRANK, M., AND MITTAL, P. Sybilbelief: A semi-supervised learning approach for structure-based sybil detection. *IEEE TIFS 9*, 6 (2014).

[12] GONG, N. Z., AND LIU, B. You are who you know and how you behave: Attribute inference attacks via users' social friends and behaviors. In *USENIX Security Symposium* (2016).

[13] GONG, N. Z., TALWALKAR, A., MACKEY, L., HUANG, L., SHIN, E. C. R., STEFANOV, E., SHI, E. R., AND SONG, D. Joint link prediction and attribute inference using a social-attribute network. *ACM TIST* (2014).

[14] GONG, N. Z., XU, W., HUANG, L., MITTAL, P., STEFANOV, E., SEKAR, V., AND SONG, D. Evolution of social-attribute networks: Measurements, modeling, and implications using google+. In *IMC* (2012).

[15] GUPTA, P., GOTTIPATI, S., JIANG, J., AND GAO, D. Your love is public now: Questioning the use of personal information in authentication. In *AsiaCCS* (2013).

[16] HE, J., CHU, W. W., AND LIU, Z. V. Inferring privacy information from social networks. In *IEEE Intelligence and Security Informatics* (2006).

[17] KOSINSKI, M., STILLWELL, D., AND GRAEPEL, T. Private traits and attributes are predictable from digital records of human behavior. *PNAS* (2013).

[18] LABITZKE, S., WERLING, F., AND MITTAG, J. Do online social network friends still threaten my privacy? In *CODASPY* (2013).

[19] LINDAMOOD, J., HEATHERLY, R., KANTARCIOGLU, M., AND THURAISINGHAM, B. Inferring private information using social network data. In *WWW* (2009).

[20] MCPHERSON, M., SMITH-LOVIN, L., AND COOK, J. M. Birds of a feather: Homophily in social networks. *Annual Review of Sociology* (2001).

[21] MINKUS, T., DING, Y., DEY, R., AND ROSS, K. W. The city privacy attack: Combining social media and public records for detailed profiles of adults and children. In *COSN* (2015).

[22] MISLOVE, A., VISWANATH, B., GUMMADI, K. P., AND DRUSCHEL, P. You are who you know: Inferring user profiles in online social networks. *WSDM* (2010).

[23] NARAYANAN, A., PASKOV, H., GONG, N. Z., BETHENCOURT, J., SHIN, R., STEFANOV, E., AND SONG, D. On the feasibility of internet-scale author identification. In *IEEE S & P* (2012).

[24] OTTERBACHER, J. Inferring gender of movie reviewers: exploiting writing style, content and metadata. In *CIKM* (2010).

[25] PEARL, J. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. 1988.

[26] SAAD, Y. *Iterative methods for sparse linear systems*. Siam, 2003.

[27] SWEENEY, L. k-anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems* (2002).

[28] THOMAS, K., GRIER, C., AND NICOL, D. M. unfriendly: Multi-party privacy risks in social networks. In *PETS* (2010).

[29] TRAUDA, A. L., MUCHAA, P. J., AND PORTER, M. A. Social structure of facebook networks. *Physica A: Statistical Mechanics and its Applications 391*, 16 (2012).

[30] WANG, G., KONOLIGE, T., WILSON, C., AND WANG, X. You are how you click: Clickstream analysis for sybil detection. In *USENIX Security Symposium* (2013).

[31] WANG, G., ZHANG, X., TANG, S., ZHENG, H., AND ZHAO, B. Y. Unsupervised clickstream clustering for user behavior analysis. In *CHI* (2016).

[32] WEINSBERG, U., BHAGAT, S., IOANNIDIS, S., AND TAFT, N. Blurme: Inferring and obfuscating user gender based on ratings. In *RecSys* (2012).

[33] ZAMAL, F. A., LIU, W., AND RUTHS, D. Homophily and latent attribute inference: Inferring latent attributes of twitter users from neighbors. In *ICWSM* (2012).

[34] ZHELEVA, E., AND GETOOR, L. To join or not to join: The illusion of privacy in social networks with mixed public and private user profiles. In *WWW* (2009).