

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 AttrInfer, 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}_{Γ_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:

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

¹We 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* $\phi_{uv}(x_u, x_v)$ for the edge (u, v) as

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

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

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

where $Z = \sum_{x_U} \prod_{u \in V} \phi_u(x_u) \prod_{(u,v) \in E} \phi_{uv}(x_u, x_v)$ 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 AttrInfer, 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 t th iteration is

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

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:

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

which is equivalent to

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

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 ATTRINFER

The basic version of AttrInfer has two shortcomings: 1) AttrInfer is not scalable enough, and 2) AttrInfer 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 AttrInfer to address these shortcomings.

5.1 Eliminating Message Maintenance

One of the major reasons why basic AttrInfer 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:

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

Considering Equation 3, we have:

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

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:

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

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

5.2 Linearizing as a Matrix Form

AttrInfer 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 AttrInfer converge via linearizing Equation 4. The resulting optimized AttrInfer 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 $\hat{\mathbf{y}} = [y_1 - 0.5, y_2 - 0.5, \dots]$.

For convenience, we denote by $\mathbf{p}^{(t)} = [p_1^{(t)}; p_2^{(t)}; \dots; p_{|V|}^{(t)}]$ a column vector all users' posterior probability in the t th iteration, and its residual vector as $\hat{\mathbf{p}}^{(t)}$. Similarly, we denote by a column vector $\mathbf{q} = [q_1; q_2; \dots; q_{|V|}]$ 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 AttrInfer 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. \square

Theorem 1 enables us to design an efficient version of AttrInfer: 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. AttrInfer 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 $\frac{\frac{1}{k} + \varepsilon_1}{\frac{1}{k} + \varepsilon_2} \approx \frac{1}{k} + \varepsilon_1 - \frac{\varepsilon_2}{k}$ 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 AttrInfer 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 AttrInfer.

THEOREM 2 (SUFFICIENT AND NECESSARY CONDITION). *The sufficient and necessary condition that AttrInfer 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 $\hat{w} < \frac{1}{2\rho(\mathbf{M})}$. \square

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 AttrInfer'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., $\rho(\mathbf{D}) \leq \|\mathbf{D}\|$, 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 AttrInfer converges is*

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

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

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

$$\iff \hat{w} < \frac{1}{2\|\mathbf{M}\|_1} = \frac{1}{2\max_{u \in V} d_u}. \quad (12)$$

□

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, AttrInfer 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 AttrInfer 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 AttrInfer. Computational time of the optimized AttrInfer 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 AttrInfer does not rely on the number of target users whose attributes an attacker wants to infer. We note that the basic version of AttrInfer 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 AttrInfer 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

Table 1: Dataset statistics.

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

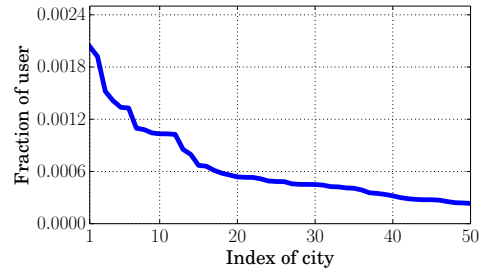


Figure 1: Fraction of users who claim a city as *cities lived*.

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.

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 AttrInfer, 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].
- **AttrInfer.** Our proposed method, which combines social graph and behaviors, as well as computes the scores for all test users

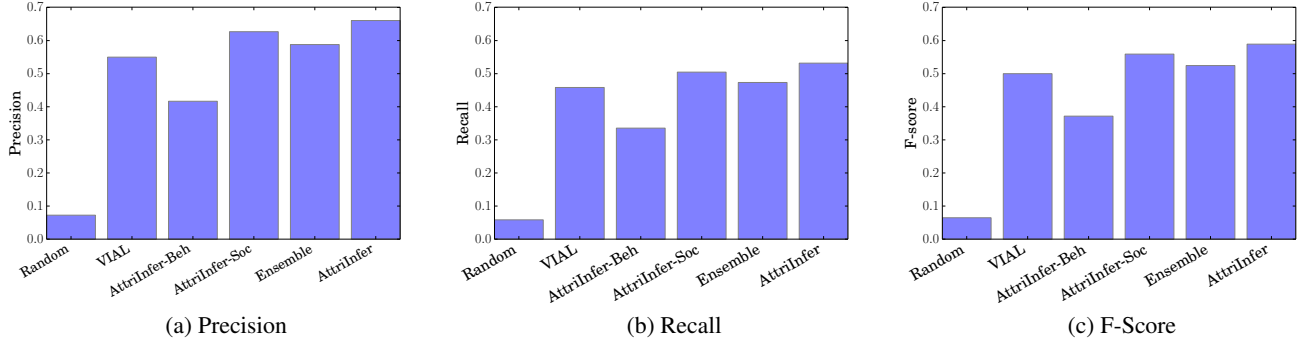


Figure 2: Precision, Recall, and F-Score of compared methods.

simultaneously. We set the residual of homophily strength (i.e., \hat{w}) as $1/(2 \times \text{average degree})$ 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., $F\text{-Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$.

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.

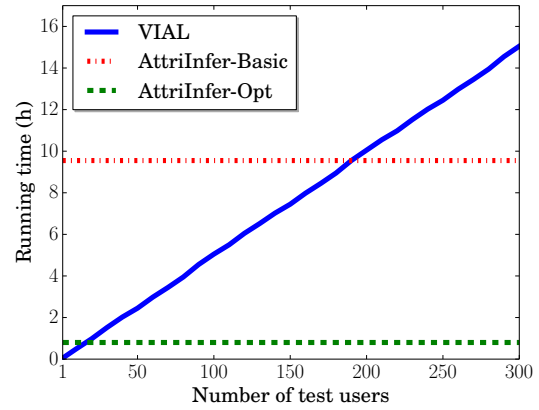


Figure 3: Computational times of VIAL, basic version of AttriInfer (AttriInfer-Basic), and optimized version of AttriInfer (AttriInfer-Opt) as we attack more users.

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.

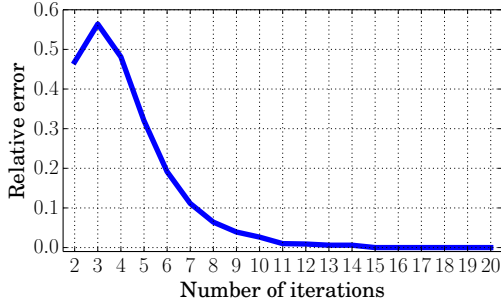


Figure 4: Relative errors of the posterior probability vectors of AttrInfer-Opt in two consecutive iterations as a function of the number of iterations.

AttrInfer is convergent: Figure 4 demonstrates that the optimized version of AttrInfer indeed converges. Specifically, we define a relative error of the posterior probability vector in the t th 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, AttrInfer 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 AttrInfer 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 AttrInfer, a new method to infer private user attributes in online social networks. AttrInfer 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 AttrInfer, we associate a binary random variable with each user; the binary random variable characterizes whether a user has a considered attribute or not. AttrInfer first learns a prior probability that a user has the attribute using users’ behaviors through a logistic regression classifier. Then, AttrInfer 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, AttrInfer uses Loopy Belief Propaga-

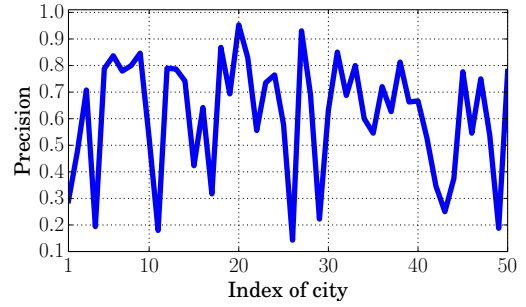


Figure 5: Precision for top-1 prediction of each city. Cities are sorted in a descending order using their populations.

tion (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 AttrInfer to be scalable and guaranteed to converge. We compare AttrInfer with state-of-the-art methods using a large-scale Google+ dataset with 5.7M users. Our results demonstrate that AttrInfer substantially outperforms state-of-the-art methods in terms of both inference accuracy and scalability. Moreover, the optimized version of AttrInfer 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 $\mathcal{Z}_u^{(t)} = q_u \prod_{v \in \Gamma(u)} m_{vu}^{(t)} + (1 - q_u) \prod_{v \in \Gamma(u)} (1 - m_{vu}^{(t)})$. By rewriting $p_u^{(t)} = \frac{1}{\mathcal{Z}_u^{(t)}} q_u \prod_{v \in \Gamma(u)} m_{vu}^{(t)}$ with the corresponding residual variables, we have:

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

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

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

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

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

Adding Equation 13 with Equation 14 yields $\ln \mathcal{Z}_u^{(t)} = |\Gamma(u)| \cdot \ln(0.5)$. 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., MACKAY, 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).