Birds of a Feather Reveal Together: A Promising Approach to Social Network De-Anonymization

Karan Kaur
kkaur89@umd.edu

Liqian Zhang
conter@umd.edu

Abstract

A number of approaches have been proposed in recent years for social network de-anonymization, utilizing either hand crafted tricks or knowledge of graph structure. We propose a machine learning based approach that does not assume any prior knowledge of the social graph's relation with the auxiliary graph. Apart from the generic degree information feature, we include combination of a variety of graph node metrics such as PageRank, Signed Spectral Ranking etc. which help us to generalize and scale better than the existing approaches. Our innovations are: 1) the ability to handle negative weighted graph with various newly applied SNG metrics and generalized `diff` expression; 2) two ScoreBoard selection algorithms to ensure robust prediction of node mappings.

1 Introduction

In recent years, more and more public datasets involving metadata of personal information have been emerging. These data resources are gathered and shared by the network owners, data holders or research communities for the purpose of customized advertising, better recommendation and conducting data mining research or contest [3].

To release the data without violating the privacy of users, the dataset has to be anonymized. Anonymization refers to protecting the privacy of users by removing the personally identifiable information (PII) such as names or demographic information. However, mere removal of PIIs is not strong enough in front of an deliberate attacker with auxiliary information of the metadata. The attacker here wants to recover or recognize the sensitive data that has been removed. This process is called de-anonymization.

Among all the datasets, we are especially interested in social network graphs (SNG). The social network graph consists of nodes and edges and information associated with each graph and edge. So, existence of an edge between two nodes can provide information about the two nodes, the information can be confidential e.g. sexual orientation. Hence, protecting the personally identifiable information is of utmost importance. The recovery of such identity is also called user identification in social media research [15]. In this project, we are trying to build a framework to analyze privacy and anonymity in social networks and a re-identification algorithm targeting anonymized social networks.
We implement a seedless machine learning model that generalizes and scales better since we incorporate multiple features of the nodes in the social graph. This does not assume any prior knowledge of the mappings between anonymized social network graph and the auxiliary graph and hence is more practical to use. With extensive features of the nodes, we are able to handle signed and directed graphs along with the undirected graphs. Another key contribution of our approach is to introduce the ScoreBoard-family selection algorithms to make our predictions more stable.

2 Background

2.1 Social Network Graph

A social network graph $\mathbb{S} = (G, \mathbb{X}, \mathbb{Y})$, where

> G = (V, E) is a direct graph with a set of nodes V and a set of relations/edges based on these nodes $E \subset V \times V$,
>
> $\mathbb{X}$ is a set of attributes depicting information for each node,
>
> $\mathbb{Y}$ is a set of attributes containing information on each edge.

In a social network, each node is a person and the edge between two nodes is their relationship. If an attacker can recognize a node v in the released data, the sensitive personal information of the node v in $\mathbb{X}$, and the information about the relationship of node v and its neighbors in $\mathbb{Y}$ is consequently exposed.

2.2 Anonymization Scheme

Anonymization of SNG is generally done in the following 3 schemes:

Clustering-based schemes: In clustering based schemes, we don't care about the relation and interplay between the nodes and hence, multiple nodes are grouped together. However, this restricts the potential researches due to the fact that we are ignoring the relation and interplay between the nodes. [4] introduces one such privacy-preserving scheme.

***k*-Anonymity based schemes**: This scheme ensures that a node is distinguishable from $k-1$ other nodes. However, it is of little use to protect high-dimensional datasets and is known to be NP-hard.

Perturbation-based schemes: Certain nodes and edges are modified so that they are not easily recoverable from the raw map. We will adopt this scheme because neither it aggregates the nodes as clustering-based does, nor it is too complex.

2.3 Attacking Scheme

Based on whether the attacker has the knowledge of some identity in the released data, there are two types of attacking schemes.

Seed-based attacks. Seeds are nodes in the graph that are known or easily recovered mappings across public data and auxiliary information with prior knowledge. The revelation of the neighboring nodes is propagated iteratively from these seeds [5].

Seedless attacks. Seedless attacks generally use structural correlation to find mappings. It doesn't require the attacker to tune parameters and invent handcrafting heuristics which are needed for the seeds. We want to employ this kind of attack by transferring the problem into a learning task of classifying a pair of nodes as identical or non-identical.

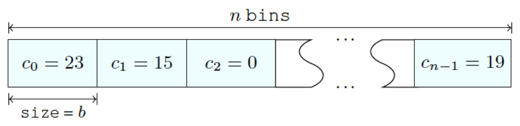
3 Related Work

3.1 Threat Model

The threat model has been fully developed and discussed in [11]. In this most widely-used and practical model, the adversary has access to an auxiliary graph which is used as the side information to re-identify individuals in a sanitized graph. The adversary is simulated by sampling overlapping graphs from a large social graph. The data holders sanitize the graphs prior to publishing them by removing all identifying information and adding noise via edge-manipulation. The sampling generates auxiliary and sanitized graphs from the original graph. The simulation process will be discussed in subsection 4.1.

3.2 De-Anonymization as Classification

A similar seedless machine learning based approach has been taken in [13] and [12] where random forests machine learning model have been used for deanonymization. The learning task is to classify a pair of nodes selected at random from G_{san} and G_{aux} as identical or non-identical. A node's feature vector has been defined using it's neighborhood degree distribution. The Neighbors of a graph node is split into two categories, 1-hop nodes and 2-hop nodes, with the shortest distance between a node and its 1-hop and 2-hop Neighbors being one and two respectively. The feature vector then consists of quantized neighborhood degree distribution. Each bin contains the count of nodes that have degree in a given range.



The model is then trained in the absence of ground truth data by splitting G_{san} and G_{aux}. For classification, the node features are extracted from the test data pair and passed through a random forest classifier with 400 estimators, which assigns probability of being identical to the node pair.

3.3 Graph Node Metrics

Apart from degree information, we can apply many graph node metrics from existing social network literature [7]. A node metric is a function $F : V \to \mathbb{R}$ that measures the importance of a node. These metrics can be applied to nodes in directed, signed weighted graphs. Here are some metrics we employ in this project:

Fans Minus Freaks (FMF)[8]. It is defined as the total positive incoming weights minus the total negative incoming weights.

PageRank (PR). PageRank [2] was originally designed to measure the importance of web pages given the reference of one page to another. The

PageRank of a node u is defined as

$$PR(u) = \frac{1-\delta}{|V|} + \delta \sum_{\{v|(v,u)\in E\}} \frac{PR(v)}{|\{w|(v,w)\in E\}|}$$

where δ is the damping factor taken to be 0.85 here.

A modified PageRank value (**modPR**) of a node v is $PR(v,G^{+}) - PR(v,G^{-})$ where G^{+} is the subgraph with only positive edges and G^{-} is the subgraph with only negative edges.

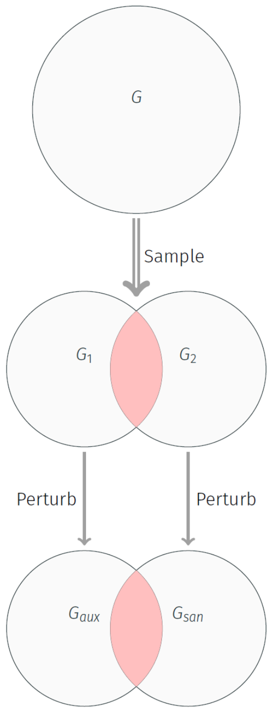
Signed Spectral Ranking (SSR). Built upon PageRank. SSR [8] is the dominant left eigenvector of the signed adjacency matrix A of the graph. Then the SSR for a node can be extracted the corresponding component from the vector.

Signed Eigenvector Centrality (SEC). Similar to SSR, SEC [1] of a node v can be grabbed from the vector x that satisfies the equation $Ax = \lambda x$, where A is the signed adjacency matrix, λ is the greatest eigenvalue.

4 Approach

4.1 Public and Auxiliary Data Simulation

For ethical issues, we use freely available and open social network datasets specifically published for research by the relevant organizations. We use Enron email exchange history dataset from Stanford Large Network Dataset Collection (SNAP) [9], for this project. Enron dataset has been widely used by previous researchers on de-anonymization experiments [6], an undirected graph G with nodes = 36,692 and edges = 183,831. We sample graph G to generate overlapped G sanitized - G_{san} and G auxillary - G_{aux}, to simulate the public data and attacker's auxiliary information respectively. This process can be illustrated by the following graph [14].



Generating perturbed graphs.

Set the node overlap parameter α_V, so for every node in the original graph, there is a chance α_V that the node will be common in V_{san} and V_{aux}. If it is not, then it will equally likely appear either in V_{san} or V_{aux} but not both.

Set the edge overlap parameter α_E, which indicates the sanitization done by the releaser, i.e. edge perturbation, as well as the incomplete or error in attacker's auxiliary knowledge. For all edges in the original graph, first project it on V_{san} and V_{aux} respectively and independently. Then at a chance of $\frac{1-\alpha_E}{1+\alpha_E}$ it will be deleted in V_{san}, at a chance of $\frac{1-\alpha_E}{1+\alpha_E}$ (same probability but independently) it will be deleted in V_{aux}.

In the Enron dataset we pick $\alpha_V = 1$ and $\alpha_E = 0.43$ as in others' work [6, 12] .

4.2 Feature Extraction

For a node v in graph G, denote Feat(v,G) to be the feature vector of v with respect to G.

To measure the importance of a node in a graph, following are the metrics as introduced in subsection 3.3:
`Metrics(v) = {Deg(v), FMF(v), PR(v), ModPR(v), SSR(v), NR(v), SEC(v), PolarityRank(v), PolarityRankNN(v)}`

Next, we define the domain of the node v, meaning nodes with following relations to node v will appear in the feature vector of node v:
`Domain(v) = {ego(v), 1-hop-Neighbors(v), 2-hop-Neighbors(v)}`
where $\texttt{ego(v)} = \{\texttt{v}\}$,
$1-hop-Neighbors(v) = \{u|(v,u) \in E)\}$,
$2-hop-Neighbors(v) = \{u|(v,w) \in E, (w,u) \in E\}$

Then the feature vector of node v, $\texttt{Feat(v)}$, contains $\texttt{avg}\{\texttt{f(u)}\}$ for every $f \in \texttt{Metric}$ and $u \in \texttt{Domain(v)}$, namely the average of the set of the operator $\texttt{Metrics(v)}$ applied on each set of $\texttt{Domain(v)}$. So for current metrics we use, there are 27 features for a node.

4.3 Pairwise Data for Classification

After we have $\texttt{Feat}(\texttt{V}_1, \texttt{G}_1)$ and $\texttt{Feat}(\texttt{V}_2, \texttt{G}_2)$, we want to generate the pairwise dataset either for training or for testing. So we have to pick a node v_1 in V_1 and a node v_2 in V_2, compare the difference between their feature vectors and feed this information into the model.

Feature Difference. To write the pair into a line of training/testing data, we first compute the feature vector pairs $\texttt{Feat}(\texttt{v}_1)$ and $\texttt{Feat}(\texttt{v}_2)$, and write the difference measurement of the pair as a field/attribute. Say the corresponding fields in the feature pair ($f_1 = \texttt{Feat}_[\texttt{v}_1][\texttt{i}],\ \texttt{f}_2 = \texttt{Feat}_[\texttt{v}_2][\texttt{i}]$). Intuitively the more similar f_1 and f_2, the higher likely the two nodes are identical. We create the difference function $\texttt{diff}(\texttt{f}_1, \texttt{f}_2) = \frac{|\texttt{f}_1 - \texttt{f}_2|}{\max(|\texttt{f}_1|, |\texttt{f}_2|)}$ based on the Silhouette Coefficient used in [12]. Our innovation is out of the fact that we use negative-value features and this variation generalizes the difference/similarity measurement to $\mathbb{R}^-$.

Finally, there is another attribute, training goal $\in \{0, 1\}$ indicating the ground truth of whether this pair is an identical pair or not, fed into the model.

PairTrain(G_1, G_2). To generate the training data, for each node in G_1, if it also appears in G_2, we write this positive pair to the training data, as well as I_n randomly selected negative cases paired with it to the training data. This has $(1 + I_n)$ lines; If it does not appear in G_2, we pair it with NI_n randomly selected negative cases and write to the training data. This has (NI_n) lines.

This will yield the positive-negative training ratio $\frac{\alpha_V}{\alpha_V \times I_n + (1-\alpha_V) \times NI_n}$.

Overall, we have 27,586 identical examples (positive cases) and 1,487,104 non-identical examples (negative cases), with parameter $I_n = 50$ and $NI_n = 30$, this is an unbalanced dataset, but we is actually more balanced than that used in [12].

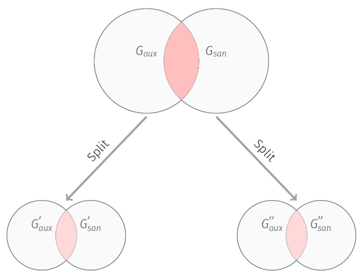
PairTest(G_1, G_2). To generate the test data, we enumerate all possible pairs of G_1 and G_2. There are $|G_1| \cdot |G_2|$ such pairs, each pair will be output in the form of feature difference. The reason for complete pairing is that for node mapping, i.e. pick the highest output from all pairs of v_1 to get its mapping node in V_2, see subsection 4.6 for information on node mapping.

4.4 Feasibility Test

In our initial attempt, we tested the feasibility of this approach by training an ideal model - ground truth is known. We had 27,586 identical examples (positive cases) and 1,487,104 non-identical examples (negative cases). We randomly shuffled and split the data int 80% train and 20% test. We trained a random forest classifier with 100 estimators. The precision for identical pair is $\frac{1965}{1965+471} = 80.7\%$. The recall for identical pair is $\frac{1965}{1965+3626} = 35.1\%$. Although we got an accuracy of $\frac{1965+296876}{1965+471+3626+296876} = 98.6\%$, it can not be regarded as the accuracy of de-anonymization since we focused too much on the prediction precision and missed many deanonymizable nodes. The F_1 score is $\frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = 48.9\%$. So though our precision 80.7% beats the 41.91% accuracy (same meaning as our precision) in the paper [12], this is an incomplete comparison with little persuasiveness due to our low recall.

4.5 Training Without Ground Truth

However the attempt above is an ideal setting and not the practical scenario. In real world as an attacker, we dont know any ground truth about the correspondence between G_{san} and G_{aux}. So, we need to split G_{san} into two overlapped sets to manually create correspondence as the ground truth of training data. Similarly for G_{aux}. The following graph from [14] demonstrates the process.



Training without ground truth by splitting the original graphs.

The attacker wants to split in the same way how G_{san} and G_{aux} are overlapped from G as much as possible (note that he doesn't know the parameters nor the correspondence). Hence, 1) the attacker needs a good estimation of the overlapping parameter α_V between his information and the released data. We choose $\alpha'_V = 0.5$ which in worst case can differ 0.5 from the real value ($\alpha_V = 1$ from subsection 4.1 so this is the worst case). 2) $\alpha'_E = 0$. Edge perturbation is unnecessary to perform again because the two splits $(G'_{aux}, G'_{san}), (G''_{aux}, G''_{san})$ originally have perturbed edges across them, which is confirmed by Sharad[1].

4.6 De-Anonymization

After we test the model with PairTest(G_{san}, G_{aux}), we'll have a confidence value for each node pair being a true mapping. i.e. for a node v_1 in G_{san}, we've evaluated the probability of it pairing to each node in G_{aux}. In this prediction array of probability (let's call it $\mathtt{PAP}(\mathtt{v}_1)$), if we just choose $\max_{u \in V_2} \mathtt{PAP}(\mathtt{v}_1, \mathtt{u})$ as the output match, the result will be unstable because our pair identical-or-not prediction error is amplified in this process.

Inspired by [10] in de-anonymizing users in Netflix by picking the most similar users from public records on IMDB, they came up with a ScoreBoard-RH algorithm, where they only output the match with highest score if the second highest score is relatively low. We modify this algorithm to best fit the property of our prediction array of probability (PAP). This is so because in [10] , ScoreBoard-RH requires that the max score is significantly higher than the second score.

ScoreBoard-THR(`THR, size`). Candidates are pair matches with certain threshold.

$$cand_{\mathtt{THR}} = \{u | \mathtt{PAP}(\mathtt{v}_1, \mathtt{u}) > \mathtt{THR}\}.$$

If $|cand| <= \mathtt{size}$, match v_1 with the one with highest probability in $\mathtt{PAP}(\mathtt{v}_1)$.

ScoreBoard-ECC(`ECC, size`). Candidates are pair matches with certain eccentricity.

$$cand_{\mathtt{ECC}} = \{u | \mathtt{PAP}(\mathtt{v}_1, \mathtt{u}) > \mathbb{E}[\mathtt{PAP}(\mathtt{v}_1)] \cdot \mathtt{ECC}\}.$$

If $|cand| <= \mathtt{size}$, match v_1 with the one with highest probability in $\mathtt{PAP}(\mathtt{v}_1)$.

[1]In his email "Well it is hard to produce exactly similar data, but given the perturbed datasets splitting graphs to produce training samples gives you something similar. Please note that perturbing G'_{aux}, G'_{san}, G''_{aux} and G''_{san} would introduce yet another change to their neighbourhood and depending upon the perturbation scheme might lower node degree too much, which would make them useless. Ideally you would like a fresh graph and then split it and then perturb each of the split graphs. Here we try to get close to that by just splitting graphs which have already been perturbed, namely G_{aux} and G_{san}."

We use the ScoreBoard-THR(THR, size) and ScoreBoard-ECC(ECC, size) on the dataset. We couldn't achieve that as those PAP score varies little and the desired match could not outstand. So the prediction is only made if the high-score-group contains less than *size* (a parameter) candidates. For example, ScoreBoard-THR(THR=0.6, size=10) will only output the highest match if there are less than ten matches with score greater than 0.6. Otherwise prediction will not be made to avoid losing accuracy. Likewise ScoreBoard-ECC(ECC=5, size=10) will only output the highest match if there are less than 10 matches with score greater than 5 times average match score.

5 Multi-Pass Experiment

The idea for node identification is to re-identify high degree nodes first and use them to identify low degree nodes. Different from [12] where they use 3-phase seedless attack (3PSL) with thresholds $t_1 = 25$, $t_2 = 9$, $t_3 = 2$, we consider only two passes with the first pass aiming at nodes with degree in the 5% highest percentile. For each pass, a specific random forest model is trained.

For simplicity, we define a subgraph function $\mathtt{HighDeg} : \mathtt{G} \rightarrow \mathtt{G}$ containing only the nodes with degree in 5% highest percentile and their corresponding edges.

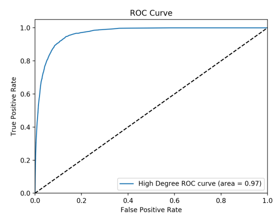
5.1 First Pass

As mentioned in subsection 4.5, we train the first-pass model with training data from graph G'_{san}, G'_{aux}, G''_{san}, G''_{aux}.

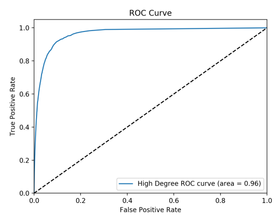
Because the model will be tested on nodes in G_{san} and G_{aux} with degree in the 5% highest percentile i.e. PairTest($\mathtt{HighDeg}(\mathtt{G}_{\mathtt{san}})$, $\mathtt{HighDeg}(\mathtt{G}_{\mathtt{aux}})$). We come up with two training scheme:

Scheme 1: Training with G'_{san}, G'_{aux} and G''_{san}, G''_{aux} :
PairTrain(G'_{san}, G'_{aux}) + PairTrain(G''_{san}, G''_{aux}).
Scheme 2: Training with High Degree Nodes in G'_{san}, G'_{aux} and G''_{san}, G''_{aux} :
PairTrain($\mathtt{HighDeg}(\mathtt{G}'_{\mathtt{san}})$, $\mathtt{HighDeg}(\mathtt{G}'_{\mathtt{aux}})$) + PairTrain($\mathtt{HighDeg}(\mathtt{G}''_{\mathtt{san}})$, $\mathtt{HighDeg}(\mathtt{G}''_{\mathtt{aux}})$).

The classification Receiver Operating Characteristic (ROC) curves for the two schemes are as follows:

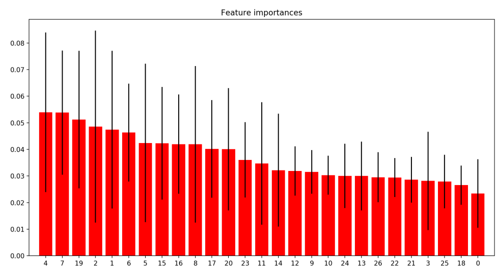


(a) ROC curve for scheme 1



(b) ROC curve for scheme 2

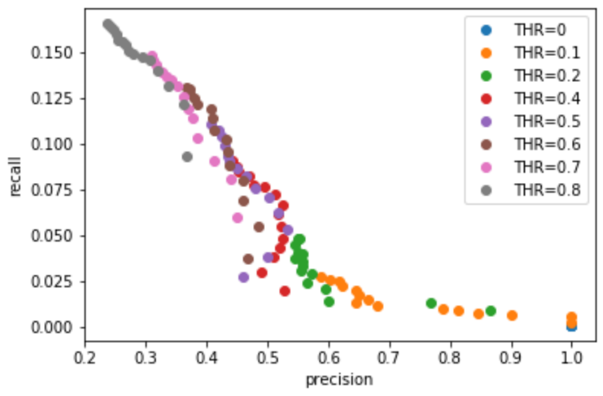
The difference between the results of the two schemes are insignificant. However, we continue the experiment with scheme 1 since scheme 2 has an added layer of work to find the highest degree nodes.



Feature ranking. Every feature contributes nearly equally. The number on the x-axis is divided by 3, if the remainder is 0 then it's a metric on ego; if the remainder is 1 then it's a metric on 1-hop Neighbors. From the first few rankings we could see 1-hop features being the most important features.

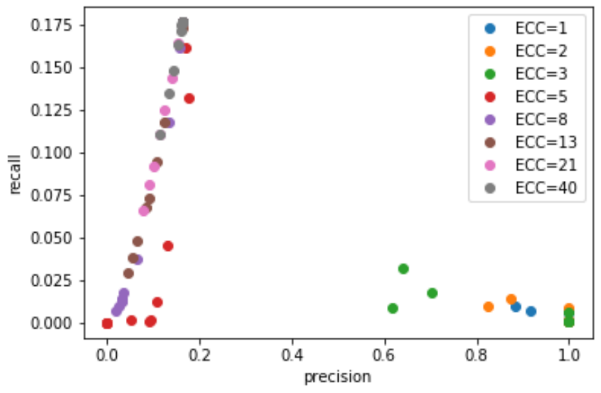
After the classification we get $PAP(v)$ for each node $v \in G_{san}$. Then we want to match pairs with high confidence like described in subsection 4.6

The graph below is the result of applying ScoreBoard-THR with different `THR` and `size`.



ScoreBoard-THR(`THR`, `size`), with `THR` $\in [0..0.8]$, `size` $\in [0..20]$. Each colored point is a feasible solution with such parameters, reaching x-axis of precision and y-axis of recall

The graph below is the result of applying ScoreBoard-ECC with different `ECC` and `size`.



ScoreBoard-ECC(`ECC`, `size`), with `ECC` being some values from $[1..40]$, `size` $\in [0..20]$. Each colored point is a feasible solution with such parameters, reaching x-axis of precision and y-axis of recall

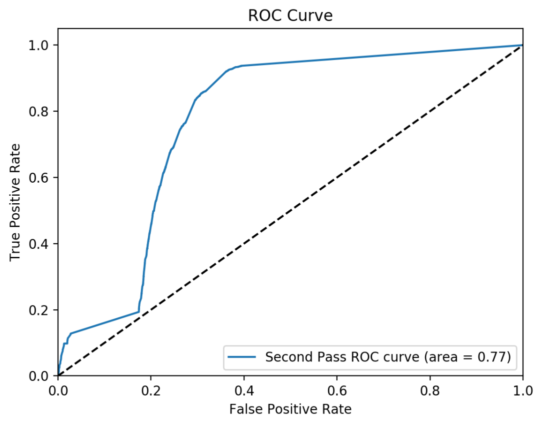
As expected, when we traverse different combinations of parameters, the precision-recall trade-off should show a graph of negative correlation. From the two figures above, we can see ScoreBoard-THR gets a better silhouette of the precision-recall trade-off. So we pick the solution (point in the ScoreBoard-THR graph) maximizing (recall+precision) to continue on the second pass. This solution will output matches with around 0.5 precision and 0.075 recall.

5.2 Second Pass

The training data for the second pass also comes from G'_{san}, G'_{aux}, G''_{san}, G''_{aux}.

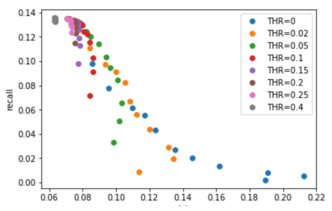
But this time for each node $v \in V'_{san} \cap V'_{aux}$, we do PairTrain$(\texttt{Neighbors}(\texttt{v}, \texttt{G}'_{\texttt{san}}),\ \texttt{Neighbors}(\texttt{v}, \texttt{G}'_{\texttt{aux}}))$. Same goes for G''_{san}, G''_{aux}.

The classification Receiver Operating Characteristic (ROC) curve for the second pass:

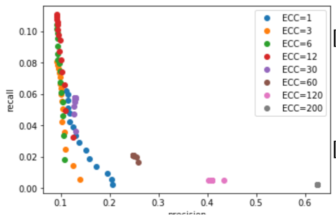


One can see that the precision drops dramatically due to error propagation: If we matched a false pair (u, v) in the output of first pass match, we will never find a match in $(\texttt{Neighbors}(\texttt{u}), \texttt{Neighbors}(\texttt{v}))$ in most cases.

The ScoreBoard-THR and ScoreBoard-ECC for second pass are shown below.



(c) ScoreBoard-THR for second pass



(d) ScoreBoard-ECC for second pass

Due to the errors of first pass prediction, as well as the heterogenity of the training/testing data of second pass, the precision and recall both are very low in the second pass phase, which means that because of the errors in first pass prediction, using the nodes re-identified in first pass phase (higher degree nodes) is not helpful to re-identify nodes in the second pass.

6 Conclusion

The seedless machine learning model we implemented generalizes and scales better since we incorporate multiple features of the nodes in the social graph. Also, since this does not assume any prior knowledge of the mappings between anonymized social network graph and the auxiliary graph, this makes it more practical to use. Furthermore with these extensive features of the nodes, we are able to handle signed and directed graphs along with the undirected graphs. Another key contribution of our approach is to introduce the ScoreBoard-family selection algorithms to make our predictions more stable.

References

[1] Bonacich, P., and Lloyd, P. Calculating status with negative relations. *Social networks 26*, 4 (2004), 331–338.

[2] Brin, S., and Page, L. Reprint of: The anatomy of a large-scale hypertextual web search engine. *Computer networks 56*, 18 (2012), 3825–3833.

[3] de Montjoye, Y., Smoreda, Z., Trinquart, R., Ziemlicki, C., and Blondel, V. D. D4d-senegal: The second mobile phone data for development challenge. *CoRR abs/1407.4885* (2014).

[4] Hay, M., Miklau, G., Jensen, D., Towsley, D., and Weis, P. Resisting structural re-identification in anonymized social networks. *Proceedings of the VLDB Endowment 1*, 1 (2008), 102–114.

[5] Ji, S., Li, W., Gong, N. Z., Mittal, P., and Beyah, R. A. On your social network de-anonymizablity: Quantification and large scale evaluation with seed knowledge.

[6] Ji, S., Li, W., Mittal, P., Hu, X., and Beyah, R. A. Secgraph: A uniform and open-source evaluation system for graph data anonymization and de-anonymization. In *USENIX Security Symposium* (2015), pp. 303–318.

[7] Kumar, S., Spezzano, F., and Subrahmanian, V. Accurately detecting trolls in slashdot zoo via decluttering. In *Advances in Social Networks Analysis and Mining (ASONAM), 2014 IEEE/ACM International Conference on* (2014), IEEE, pp. 188–195.

[8] Kunegis, J., Lommatzsch, A., and Bauckhage, C. The slashdot zoo: mining a social network with negative edges. In *Proceedings of the 18th international conference on World wide web* (2009), ACM, pp. 741–750.

[9] Leskovec, J., and Krevl, A. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

[10] Narayanan, A., and Shmatikov, V. Robust de-anonymization of large sparse datasets. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on* (2008), IEEE, pp. 111–125.

[11] Narayanan, A., and Shmatikov, V. De-anonymizing social networks. In *Security and Privacy, 2009 30th IEEE Symposium on* (2009), IEEE, pp. 173–187.

[12] Sharad, K. Change of guard: The next generation of social graph de-anonymization attacks. In *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security* (2016), ACM, pp. 105–116.

[13] Sharad, K. Learning to de-anonymize social networks. Tech. Rep. UCAM-CL-TR-896, University of Cambridge, Computer Laboratory, Dec. 2016.

[14] Sharad, K. A machine learning approach to social graph de-anonymization. http://ksharad.com/files/slides/slides_aisec_2016.pdf, October 2016. ACM Workshop on Articial Intelligence and Security, Vienna, Austria (AISec 2016).

[15] Zafarani, R., Tang, L., and Liu, H. User identification across social media. *ACM Trans. Knowl. Discov. Data 10*, 2 (Oct. 2015), 16:1–16:30.

List of Abbreviations and Symbols

SNG	social network graph
G_{san}	sanitized data publicly released
G_{aux}	auxiliary data in attacker's knowledge
$\texttt{Neighbors}(\texttt{v},\texttt{G})$	set of nodes in graph G having an edge from v, G will be omitted if obvious in context
$\texttt{Feat}(\texttt{v},\texttt{G})$	feature vector of node v w.r.t graph G
$\texttt{PAP}(\texttt{v})$	the arrays of probability output by our model for a given node v in G_1 with every node in G_2
$cand_{\texttt{THR}}$	the set of candidate nodes filtered from ScoreBoard-THR
$cand_{\texttt{ECC}}$	the set of candidate nodes filtered from ScoreBoard-ECC