De-anonymization based on social media user-app behavior

Rishab Kinnerkar
Department of Computer Engineering
Iowa State University
riskin@iastate.edu

Agnes Ruyin Zhao
Department of Computer Engineering
Iowa State University
ruyin@iastate.edu

***Abstract*— Social media privacy is of utmost concern to users. While social media companies take efforts to maintain the privacy of their users, the users are still vulnerable to privacy-breach related attacks. One such category of attacks is information inference attacks in which the attacker uses publicly available information to deduce user information via machine learning. Social media companies usually make user information public such that it is not possible for attackers to easily de-anonymize the corresponding user. In this paper, we explore for possible relation between two seemingly unrelated publicly available user parameters. The first parameter is user-friendship and the second is user-app rating. From our experiments we are able to show that these parameters have a connection which can be detected by machine learning methods. Thus, we conclude by explaining how it is possible for attackers to use the above-mentioned parameters to strengthen their inference attack model.**

***Keywords*— attribution inference, de-anonymization, data privacy, supervised learning, graph embeddings.**

I. INTRODUCTION

User information containing datasets are released by companies for reasons like research or policymaking. User privacy is of utmost concern while releasing such datasets and thus when datasets are released by companies, they are done so with a certain amount of obscurity in them to make linking the data to the original user a difficult task. A lot of research work has been done in this field. Social media information is vulnerable to these types of attacks and it is also possible to launch inference attacks on social networks with a mixture of non-sensitive attributes and social relationships as explained in *Cai et al.* [1]. Thus, it is of utmost importance for social media companies to realize whether the information they are making publicly available of an user could be vulnerable to an inference attack.

Existing research done in this field shows how it is possible to infer private user attributes from social networks. In Zheleva et al. [2] they make use of parameters like friendship links and group affiliations to demonstrate attribute inference. They can achieve a considerable accuracy while inferring sensitive user attribute. In our research we check to see whether user-app rating data could potentially be used in inference attack models.

We use User-friendship data and User-app data from Google Plus social media. We process this data for a certain sample of users and get vector embeddings of users with respect to their social network and another with respect to the apps they have rated. Then we train a classifier using two methods. In the first method we define a linear loss function and use the Stochastic Gradient Descent algorithm to optimize the parameters. After that we use this loss function to calculate the probability of two nodes in the different graphs to be the same user. Then we pick the predicted node as the one which is giving us the highest probability. In the second method we randomly sample users from both the dataset and label the cross-product set of their feature vector as either ‘1’ or ‘0’, depending on whether the vector embeddings belong to the same user (‘1’) or not (‘0’). Then this dataset is trained and tested using different machine learning algorithms. Our experiment results go to show that there is a correlation between these two parameters, and we conclude that User-app data could be used in an attribute inference model.

II. RELATED WORK

There has been a lot of existing research work done in this field. But to our knowledge there isn’t any published work pertaining to user-app data being used for attribute inference.

In Cai et al. [1] they demonstrate on how to launch inference attacks to exploit social networks. Their research work is broad, and they have explained theoretically how data attribute inference can be carried out. Furthermore, this research work shows and validates how social networking datasets can be processed and then classified using machine learning algorithms.

In Zheleva et al. [2] friendship links and group affiliation are used to show how inference attribution can be carried out. Their work also provides a method for extracting relevant information from large groups to get an accurate user inference attribute. Their results go to show that it is easy to infer private information from group membership data.

In Narayan et al. [3] de-anonymization has been carried out of Netflix users based on "quasi identifier" attributes. These "quasi identifier" attributes are publicly available attributes and in our research user-app data would be classified as a quasi-identifier.

Hamilton et al. [4] provides a review of the performance and states the case to why we should use low dimensional embeddings when it comes to graphs as opposed to the old method of user-defined heuristics for feature extraction. Specifically, it gives details about Large-scale information network embedding (LINE) algorithm which we have used in our research.

III. METHODOLOGY

In this section we explain our methodology. There are two methods we used for our experiment. The initial steps of our experiment are common. We have made use of the LINE algorithm for deriving the vector embeddings for our User-friendship and User-app graph. In our first method we have made use of the Stochastic Gradient Descent Algorithm (SGD) for calculating similarity scores. In our second method we have used well-known binary classifier algorithms.

A. LINE ALGORITHM

Large-scale information network embeddings (LINE) is a network embedding method which can embed large information network into vectors in low dimension [5]. It provides a way to combine the first order and second-order proximities of a graph. The first-order proximity refers to the local graph neighbourhood of a node. The second-order proximity indicates the global positions of nodes in the graph with considering two-hop adjacency neighbourhoods. With the first order and second-order proximity, we can effectively and respectively capture the local and global network structure of a large graph. LINE algorithm has shown to quickly derive accurate vector embeddings for large social networks. These vector embeddings would then be used by us for analysing our data by making it into a supervised classification problem.

B. GRADIENT DESCENT ALGORITHM

Gradient Descent is an optimization algorithm used for minimizing functions [6]. It works by iteratively taking steps to the direction of steepest descent as defined by the negative of the gradient. Depending on the model an error function is pre-defined. The gradient of the error function with respect to each parameter is then computed and the parameters are modified along the downhill direction of the gradient in order to reduce the error [7]. Let E(w) denotes the error function, in which w is a vector representing all the weights in the network, the simplest gradient descent algorithm works by modifying the weights at time step t according to: $\triangle w_t = -\varepsilon \nabla_w E(w_t)$, where ∇w represents the gradient operator according to the weights, and ϵ is the learning rate and tend to be a very small and positive number[7]. In this project, we use gradient descent to update the parameters of our model.

IV. EXPERIMENTATION

In our experiment we used the following data as shown in Fig.1.

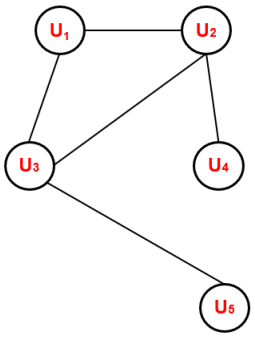
# User	# App	# Link		Avg. Degree	
		User Social Graph	User App Graph	User Social Graph	User App Graph
18928	4999	154428	831023	8	44

Fig. 1 Shows the number of users and apps we have used in our experiment. The #Link and Avg.Degree gives information on the relation between the user and app dataset.

In this research, we used a dataset with 18,928 users and 4,999 apps. The number of links in user social graph is 154,428, which means on average each user has 8 friends. The number of links in user app graph is 831,023, which means on average each user has rated 44 apps.

A. DATA PROCESSING

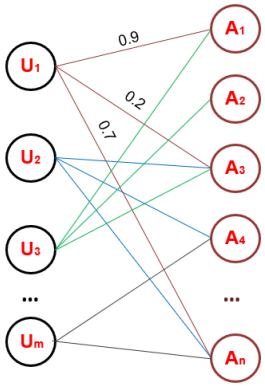
Our data consists of 18928 users and 4999 apps which users may or may not have rated. The app rating is on a scale of 0 to 1 in multiples of 0.2 where 1 indicates the highest rating for the app. The users were derived from a large social network of about 30 million users. The derived users were got in a way such that they didn't lose their social consistency. From these derived users we set them up such that they were labelled from 1 to 18928 and the rest of the apps were re-labelled from 18928 onwards to 23926. From these we set the user and app data to be processed to make it in a graph format so that we could derive their vector embeddings. The User Social graph as seen in Fig.3 is undirected and has a weightage of 1 which indicates that the connected nodes are friends in the social network.



User Social graph

Fig. 2 Shows the User Social graph where node Ui indicates a user and a connection between the nodes indicates that the two users are friends.

The user-app graph as shown in Fig.3 is a bipartite graph where the Ui node is a user and the Ai node are an App. The connected weights between elements from U and V indicate the app rating which that user has given to that app. An absence of connection between U and V indicates that the app has not been rated by the user.



User App Graph

Fig. 3 Shows the User App graph where node Ui indicates an user, node Ai indicates an app, and the connected weight in between them indicates the rating the user has given the app.

We use LINE (Large-scale information network embeddings) algorithm to embed the user social graph and user-app graph. We derive vector embeddings of each user corresponding to the User's social graph and the User's app graph. The vector embeddings are of 120 dimensions.

B. CLASSIFICATION

With the processed dataset, we split the users into training dataset and testing dataset. For this, users labelled 1 – 500 are assigned for training and user's numbered 501 – 18928 are assigned for testing.

Training dataset

For training users, we take the cross product of corresponding embedding vectors from the User-friendship graph and the user-app graph and label this as '1'. Then for each of the users in the training dataset we randomly sample other users and take their cross product and label this as '0'. The positive to negative samples are in the ratio 6:1. Now that we have our training dataset, we proceed using two methods.

Note: We experimented with different training datasets of users and ratios of positive to negative samples.

Method 1

We define a loss function and train it with features and labels we input. Simultaneously, we use the gradient descent algorithm to optimize the parameters. In order to make the optimization better, we increase the number of epochs in our model. For the rest of 18428 users, we use two "for loops" to do the cross product between user i in user-friendship graph and all the 18428 users in user-app graph, wherein the features are multiplied together. And then we feed the result into our model and get labels. For user i, we rank the second column of label and choose the highest one, get the index and compare it with i. If this index is equal to i, then we can say that the prediction is correct, otherwise it is incorrect. Then the number of correct predictions is divided by all the number of all the samples and get the accuracy.

Method 2

The training dataset used by us is fed into a machine learning API implemented in Matlab. Essentially, we have reduced this to a binary classification problem. We test for the accuracy of the classification against 23 different classification algorithms.

V. RESULTS AND DISCUSSION

Method 1

After we tested for the 18428 remaining users our algorithm was able to accurately match 4 users in our best case. The probability that a random guesser can do this matching is less than 2%. Which goes to show that these two datasets have a correlation.

Method 2

Since this is a binary classifier, we would expect a random guesser to give us an accuracy of 50%. However, after we tested and trained our datasets with different samples, we observed to be getting classification results in between 50-55%. Fig.4 shows an example of the accuracy we got.

Algorithm	Accuracy
Fine Tree	52.90%
Medium Tree	50.00%
Coarse Tree	52.30%
Linear Discriminant	52.10%
Quadratic Discriminant	52.00%
Logistic Regression	52.10%
Linear SVM	51.90%
Quadratic SVM	52.00%
Cubic SVM	52.90%
Fine Gaussian SVM	50.20%
Medium Gaussian SVM	**53.90%**
Coarse Gaussian SVM	51.90%
Fine KNN	52.00%
Medium KNN	51.40%
Coarse KNN	51.50%
Cosine KNN	52.10%
Cubic KNN	50.90%
Weighted KNN	51.30%
Boosted Trees	48.80%
Bagged Trees	50.30%
Subspace Discrimininant	53.10%
Subspace KNN	51.80%
RUSBoosted Trees	49.20%

Fig.4 Shows the different accuracy gotten from different algorithms. As we can see the Medium Gaussian SVM has an accuracy of 53.90%. On average the accuracy of all these classifiers for this run is 51.59%

Normally, after repeatedly performing this with different samples we would expect the result to converge for a random guesser to 50%. However, it didn't in our case and was above 50%. We tested the probability of getting such results randomly by

using a binomial calculator and it was less than 3%. In some cases, it was even less than 1%.

Overall from the results we got we believe that these two datasets are correlated. Our results aren't in support of a very strong correlation and we think there is a lot more scope of improvement in our experimentation. By doing the following in the future we could possibly improve the accuracy of our models and make a stronger case for our problem statement -:

1) Use a larger dataset and this would have to be done a higher processing hardware. Our personal laptops took about 5 hours to data process the user-app graph for the 18928 users. Essentially, the format in which the user data is in would require number of computations of the order = Number of users * Number of apps. Now using a larger dataset would most likely also result in more appropriate vector embeddings.

2) The LINE algorithm we use is known to perform well for very large social graphs. Provided the suggestion given above is redundant and that consistency is actually preserved when we derive users from the original data of 30 million nodes then looking into other vector embedding algorithms like Node2vec or GraREP could also help in improving the accuracy of the model.

VI. CONCLUSION AND FUTURE WORK

Through this project we improved our data processing skills and got a good understanding of run-time complexities when it comes to processing large datasets. Specifically, we learned and got a good command over Python Numpy and Pandas library. We learned to use basic tensorflow and coding specific to machine learning. While working on this paper we read a lot of other research work pertaining to specific topics in machine learning like attribute inference, vector embeddings, supervised learning, binary classification etc. which improved our overall knowledge and skills.

Our goal was to find whether user-app data could be used in attribute inference models by attackers for de-anonymization purposes. From our experiments it does seem that the user-app data is correlated with the social data of the user. Which in turn would mean that it could be used as a parameter to improve the de-anonymization accuracy. The machine learning classification results gotten by us have a less than 3% chance for a random guesser to obtain. Which all in all goes to show that these two datasets have a correlation, although small, and could be used as a parameter for improving the accuracy of attribute inference models.

In this research we used LINE algorithm to get vector embeddings from both the graphs. Now, it is known that LINE works very well for large datasets but for small datasets it might not perform as well as other algorithms like GraREP or Node2Vec. Secondly, the 18928 users which were derived were from a 30 million user dataset. It is possible that this derivation could have impacted the end accuracy. Based on our initial results for future work we would wish to try out different vector embedding algorithms on a larger user dataset with higher processing power to possibly get more accurate results.

ACKNOWLEDGEMENT

We would like to acknowledge Jinyuan Jia and Professor Neil Gong for providing us the guidance without which it would not have been possible for us to proceed with this research.

REFERENCES

[1] Cai, Zhipeng, et al. "Collective data-sanitization for preventing sensitive information inference attacks in social networks." IEEE Transactions on Dependable and Secure Computing 15.4 (2018): 577-590.

[2] Zheleva, Elena, and Lise Getoor. "To join or not to join: the illusion of privacy in social networks with mixed public and private user profiles." Proceedings of the 18th international conference on World wide web. ACM, 2009.

[3] Narayanan, Arvind, and Vitaly Shmatikov. "Robust de-anonymization of large datasets (how to break anonymity of the Netflix prize dataset)." University of Texas at Austin(2008).

[4] Hamilton, William L., Rex Ying, and Jure Leskovec. "Representation learning on graphs: Methods and applications." arXiv preprint arXiv:1709.05584 (2017)

[5] Tang, Jian, et al. "Line: Large-scale information network embedding." Proceedings of the 24th international conference on world wide web. International World Wide Web Conferences Steering Committee, 2015.

[6] Barzilai, Jonathan, and Jonathan M. Borwein. "Two-point step size gradient methods." IMA journal of numerical analysis 8.1 (1988): 141-148.

[7] Qian, Ning. "On the momentum term in gradient descent learning algorithms." Neural networks 12.1 (1999): 145-151.