Final Report of Computer Network Project

Lu Yan, Yufan Cai

June 2019

1 Introduction

1.1 Motivation

Recently, a new type of attack, coined Sybil Account comes out targeting at social networks. Sybil accounts are accounts that post fake reviews to a certain store or service in campaigns and get paid. People like store owners hire Sybil users to do this in order to promote their stores' reputation or discourage rivals. This does harm to the whole market and other customers, though. Furthermore, the attacks are always organized, so there exists some elite sybil accounts, who organize the attacks and have more complicated reviews.

Many measures have been proposed to detect Sybil accounts ever since. Generally, they can be divided into two categories: by studying user's temporal behaviors OR relationship graph centered at a user. Also, people sometimes combine these two directions. However, since elite sybil accounts have confused and deceptive reviews and features, traditional method always fail into a dilemma when considering the features of elite sybil.

In this project, we found two currently popular models GNN and GAN can be utilized to address this problem. GNN can exploit deep structural information from networks, while GAN can promote detection precision in a game playing method. The combination of GNN and GAN may lead to better performance of Sybil detection.

1.2 GNN with varying edge weight

Graphs, generally, are a representation which supports arbitrary (pairwise) relational structure, and computations over graphs afford a strong relational inductive bias beyond that which convolutional and recurrent layers can provide. Neural networks that operate on graphs, and structure their computations accordingly, have been developed and explored extensively for more than a decade.

For social network, each node are closely interrelated with their neighbourhoods, so relationships between node and node are very important when considering the feature of one node. We can utilize the topology information learned by the GNN, to let one node get the information from all the neighborhoods.

In the meantime, the relationships between two nodes sometimes contains noisy. For example, one sybil users may randomly choose some users to follow and one benign users can be followed by any users. So, when classify whether one user is sybil or not, we should reconsider its relationship. That means the edge weight will also be changed in the training process.

1.3 GAN

GAN is proposed by Goodfellow (1) in which two models are simultaneously trained: a generative model G that captures the data distribution, and a discriminating model D that estimates the probability that a sample came from the training data rather than G. This framework corresponds to a minimax two-player game. The training procedure for G is to maximize the probability of D making a mistake.

2 Dataset

There are many datasets about social network in the Internet. But unfortunately, there are few social networks that have labels showing whether one user is real user or not. Because of the privacy policy, we can not get the tweets posted by users, which needs an app-developer permission on Twitter. In that case, we choose a mixture dataset that combines a followers-friends social network and in the meantime, another dateset including labels of some users. We find that the every twitter users has his own twitter id, which is unique and unchangeable. So we can combine the network and labels through the unique twitter id to get a social network with labels. We also survey on the top conference's papers, some dataset website and information from assistant teachers. From assistant teachers, we achieved a dataset on Dianping that contains both network and labels. We specified as follows:

2.1 Twitter

We find two social networks. First, the followers-friends social network is achieved from SNAP Datasets website, Stanford Large Network Dataset Collection. This dataset consists of *circles* (or *lists*) from Twitter. Twitter data was crawled from public sources. The dataset includes node features (profiles), circles, and ego networks. It has 81,306 nodes, 1,768,149 edges. Second, USER-FOLLOWER social network is achieved from What is Twitter, a Social Network or a News Media?, Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. We combine the two social network into one social network with the unique twitter id to get a more comprehensive social network.

The user-labels dataset is achieved from POISED: Spotting Twitter Spam Off the Beaten Paths. This dataset comes from Twitter API. The API provides a stream of random users. It used a sample of 300 of them, called "seeds," and crawled the time lines for them as well as their friends and followers. Thus, it obtained data for 300 neighborhoods in Twitter, where a neighborhood consists of a seed with all his friends and followers. In the random selection of seeds, it doesn't collect data for users with more than 2,000 followers or friends, so that the crawling process would be of manageable size. It also limited the crawl to users having specified English as their language, so the further text analysis would be performed on tweets of a single language. Then, it applies four-gram analysis ti detect similar messages in every neighborhood. In total, the dataset has 1,219,991 groups of similar mes- sages with the size range being between 2 and 94,382. To evaluate the probabilistic models for spam detection, we need to have a ground-truth dataset including both spam and benign messages. It picked the top 5000 groups after ordering them by their size, and labeled these groups. The tweets in this dataset were manually checked by a group of 14 security researchers who labeled them independently.

2.2 Dianping

The Dianping is the most popular URSN in China. The dataset was crawled on Dianping from January 1, 2014 to June 15, 2015 and includes 10,541,931 reviews, 32,940 stores, and 3,555,154 users. However, this dataset doesn't specified the labels of the users. We do some prepossess specified in the part 4. Then we find that, of all the reviews, more than 108,100 reviews are fake reviews, which were generated by 21,871 regular Sybil users and 12,292 elite Sybil users. In our model, we regard the Sybil users and elite Sybil users as the fake users and the other users as real users. We separate the dataset with 80% to train our neural network and 20% to evaluate our models.

2.3 Cora

This dataset consist of 2708 scientific publications classified into one of seven classes. The citation network consists of 5429 links. Each publications in the dataset is described by a 0/1-valued work vector indicating the absence/presence of the corresponding word from the dictionary. The dictionary consists of 1433 unique words. We use this dataset for validation the effectiveness of our model.

3 related work

Tradition method for sybil detection always either the relationship network or the single feature of the node. The method in Zheng's work(2) is to use the idea that the elite users will be actively the same time with the nomal sybil users. So they detect a time interval window in which time the normal malicious users are very active. Then they scrutinize the users in this window to find out the elite users. But this method have two weaknesses. First, they may detect some benign users as elite sybil users, because these benign users are active in this window by accident. What's more, they may miss some elite sybil users who are not active in the same widow with normal sybil users. This method also ignore the information from the relationship among the users. Besides, in NDSS 2019, Binghui Wang et.al propose a general graph-based model can be used in Sybil detection (3).

4 System Overview

In this project, we propose a novel three-tier Sybil account detection system that utilizes both Graph Neural Networks (GNNs) and Generative Adversarial Networks (GANs).

In the first tier, the original data are represented in the form of undirected graph. We then apply Machine Learning with three types of features to classify communities.

Next, we create a GNN to learn and store information of these graphs. The graphs' information and their corresponding labels are fed into GAN in the third tier. The generator (G) in GAN tries to imitate the distribution of information given a certain label and generate mock information vectors. The discriminator (D), on the other hand, is trained to distinguish graph information from real-world data and that generated by G.

The system can be represented by fig 1.

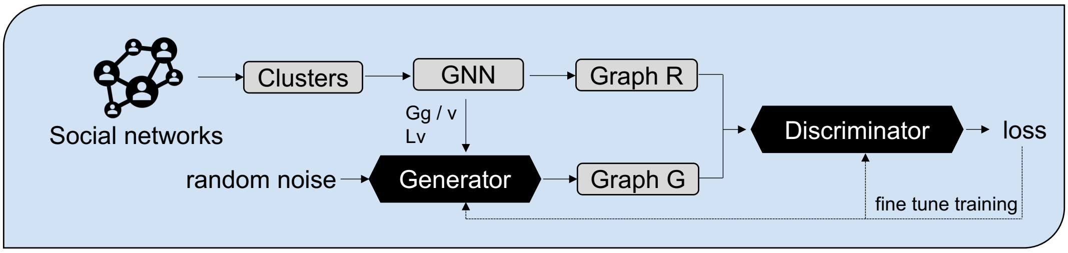


Figure 1: The System Overview

In this game playing mode, the information generated by G becomes more and more close to reality, while D's discriminating ability goes stronger and stronger. Finally, D can be adopted to detect real-world Sybil accounts.

5 Preprocessing

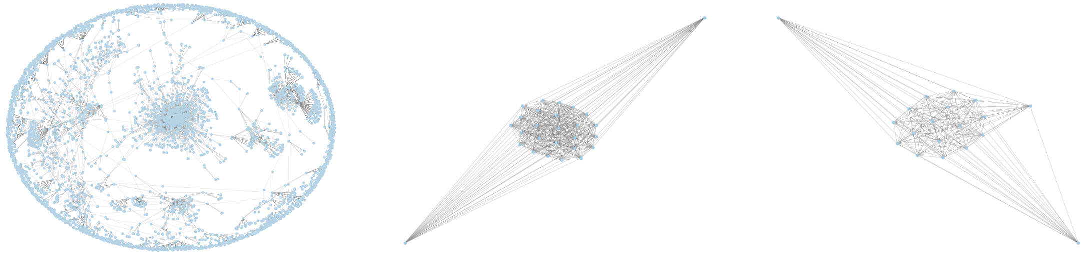
Since our work depends on both network structure and features for classification, the first step is to construct networks and choose suitable features. This part introduces our data preprocessing.

5.1 Sybil Database

As mentioned above, this dataset is obtained by Zheng. Originally, the community sub-dataset contains 4 columns, namely, user id, following users of this user, the number of all his/her following users, and the manually checked label (Sybil or Non-Sybil). The other sub-dataset, called user includes features of each user. We pick up the integer-represented features for usage, such as commentNum (the number of comments he ever published), photo (the number of photos he uploaded), article(the number of recommending articles he wrote), signIn (the number of places where he signed in), focus (the number of users he focused on), fans (the number of users following him). We use the information to create three files about the following relationships between users, the label of the users (same as the label of the community to which he belongs), and the features of the user given his/her label, respectively.

The visualization results of all the communities, one Sybil community and one benign community are showed in fig.2.

We can see from the figure that lines distribute more evenly between core users in Sybil communities, revealing closer connections inside.



(a) Graph structure of all communities (b) Graph structure of a Sybil community (c) Graph structure of a benign community

Figure 2: Visualization of Sybil Dataset

5.2 Cora Dataset

We utilize the cites file in Cora dataset to construct network. And the membership of each word in each paper serves as features of that paper. Similarly, we visualize the Cora dataset as shown in fig.3.

Figure 3: Visualization of Cora Dataset

6 Model Constructing

The core part of our project includes exploiting GNN to learn and store social network information of each class and training a generative adversarial network. We define a relationship network G = (V,E) where V represents the users and E represents the relationship between users. The objective of Gan is to learn the following two models:

$G(v_c|N(v_c), \theta_G)$, where $N(v_c)$ represent the set of neighbors of v_c. The generator tries to approximate $p_true(v_c)$, which represent the possible of malicious users for v_c $D(v_c, N(v_c), \theta_D)$ which, aims to discriminate the truth of the subgraph.

More specified details are as follows.

6.1 GNN

The GNN model is used as an encoder part. We want to use the information of the neighbourhood of one user to make the input of the GAN more comprehensive. There are two important thing: Firstly, we use several feature as the input initial feature, which is specified in section 1. Secondly, we should consider the edge weight in the graph. In many other mthods, the weights always be the same in the training process. However, one sybil can follow any users and one benign users can be followed any users. The edge weight matrix should also be learned in the training process to make the edge propagation information more correctly. For initial input of the GNN, we apply eight features with respect to three types.

6.1.1 Input Features

(a) Community-based features. There are four types of Community-based features: score deviation, reviews per store, entropy of chain stores, and entropy of districts of stores. To achieve the Sybil tasks, score deviation of reviews posted by Sybil users will become larger. Entropy of the number of reviews in each chain stores is the expected value of information contained in each of the chain stores by measuring the number of reviews occurred. We use this feature because some Sybil users post reviews only in chain stores. Entropy of districts of stores is a location-based feature to characterize mobility patterns of Sybil users that are driven by Sybil tasks. We therefore use Entropy of districts of stores to show this difference.

(b) Similarity-based network features. We redefine the network via Sybil social community construction since benign and Sybil communities have remarkable differences with respect to the graph structure. We use Average similarity and Global clustering coefficient to show the difference according to the redefined graph structures. Average similarity is the average similarity between pairwise users in a community. Sybil users in a Sybil community are assigned tasks for similar stores, but users in a benign community randomly choose stores to post reviews. Hence, similarity values between Sybil users are greater than those between benign users. Global clustering coefficient is used to measure the degree in which nodes in a graph tend to cluster together. Sybil users have the characteristics of team working, so they are more likely to be clustered together.

(c) User-based features. Since community-based features may lose information of users, we then abstract the user-based features of each user and aggregate them as a feature of the community. By analyzing Sybil communities, we observe that some Sybil users will repeatedly post reviews in the same store. We therefore define two features, Unique reviews ratio and Maximum number of duplication, to reflect this user-level behavior. Lastly, we do not use linguistic or contextual features because these features are not so effective in the URSN setting.

6.1.2 Graph network block

We now present our graph networks (GN) framework, which defines a class of functions for relational reasoning over graph-structured representations. Within our GN framework, a graph is defined as a 3-tuple G = (u,V,E) (see Box 3 for details of graph representations). The u is a global attribute; for example, u represent the Similarity-based network features, such as Average similarity and Global clustering coefficient. The $V = \{v_i\}_{i=1:N^v}$ is the set of nodes, where each v_i is a node's attribute. For example, V represent each users, with attributes for User-based features such as Unique reviews ratio and Maximum number of duplication. The $E = \{(e_k, r_k, s_k)\}_{k=1:N^e}$ is the set of edges, where each e_k is the edge's attributes, r_k is the index of the receiver node, and s_k is the index of the sender node. In our model, E is all the same, so we don't consider the E.

A GN block contains three update functions ϕ and three aggregation functions ρ,

$\widehat{e_k} = \phi_e(e_k, v_{r_k}, v_{s_k}, u)$

$\widetilde{e_i} = \rho^{e \to v}(E_i)$

$\widehat{v_i} = \phi_v(\widetilde{e}_i, v_i, u)$

$\widetilde{e} = \rho^{e \to u}(E)$

$\widehat{u} = \phi_u(\tilde{e}, \widehat{v}, u)$

$\widehat{v} = \rho^{v \to u}(V)$

where $E_i = \{(\widehat{e_k}, r_k, s_k)\}_{r_k=i,k=1:N^e}$, $\widehat{V} = \widehat{v}_{ii=1:N^v}$, and $\widehat{E} = \cup_i \widehat{E_i} = \{(\widehat{e_k}, r_k, s_k)\}_{k=1:N^e}$.

The ϕ^e is mapped across all edges to compute per-edge updates, the ϕ^v is mapped across all nodes to compute per-node updates, and the$\phi^u$is applied once as the global update. The ρ functions each take a set as input, and reduce it to a single element which represents the aggregated information. Crucially, the ρ functions must be invariant to permutations of their inputs, and should take variable numbers of arguments.

6.1.3 Computational steps

When a graph, G , is provided as input to a GN block, the computations proceed from the edge, to the node, to the global level. Algorithm below shows the following steps of computation:

1. ϕ^e is applied per edge, with arguments e_k, v_{r_k}, V_{s_k}, u, and returns $\widehat{e_k}$. In our model, this correspond to the circulation of the information from neighbors of one user. The set of resulting per-edge outputs for each node, i is $\widehat{E_i} = \{(e_k, r_k, s_k)\}_{r_k=i,k=1:N^e}$. And $\widehat{E} = \cup_i \widehat{E_i} = \{(\widehat{e_k}, r_k, s_k)\}_{k=1:N^e}$ is the set of all per-edge outputs.

2. $\rho^{e \to v}$ is applied to $\widehat{E_i}$, and aggregates the edge updates for edges that project to vertex i, into $\widetilde{e_i}$, which will be used in the next step's node update. In our model, this correspond to summing all the information from the other user.

3. ϕ^v is applied to each node i, to compute an updated node attribute, $\widehat{v_i}$. In our model, ϕ^v compute something like updated state, features of each users. The set of resulting per-node output is, $\widehat{V} = \{\widehat{v_i}\}_{i=1:N^v}$.

4. $\rho^{e \to u}$ is applied to $\widehat{E}$, and aggregates all edge updates into $\widetilde{e}$, which will then be used in the next step's global update.

5. $\rho^{v \to u}$ is applied to $\widehat{V}$, and aggregates all node updates into $\widetilde{v}$, which will then be used in the next step's global update. In our model, $\rho^{v \to u}$ compute the total feature of the network, and get the Community-based features.

6. ϕ^u is applied once per graph, and computes an update for the global attribute, $\widehat{u}$. In our model, ϕ^u computes something analogous to the Similarity-based network features, such as Average similarity and Global clustering coefficient.

6.1.4 Learning Edge weights

In the initial relationship network, the edge weight is all the same. For users who have friendships, the edge weight will 1 and for users who have no relationships, the edge weight will be 0. One innovation intuition (4) is that the edge weights and the node's label should be consistent. That means, the weights should be satisfy the following two goals:

First Goal: the GNN propagation final results should be consistent with the training labels. Second Goal: the edge weight and the GNN propagation final results should be also consistent. In particular, an edge weight is to be positive if the nodes are to have the same label, otherwise the edge weight should be approximate to zero, which means this edge will not propagate any information because two nodes are not the same label.

6.1.5 Quantifying target function

Given the training dataset L, we quantify First Goal as finding an edge weight matrix **W** such that the difference between the posterior reputation scores of the labeled nodes and their labels is minimized. Formally, we aim to find an edge weight matrix **W** that minimizes the following function:

$$L(\mathbf{W}) = \frac{1}{2}\sum_{l \in L}(p_l - y_l)^2$$

,where p_l is the predicted result of the labeled nodes, where $y_l = 1$ if label is positive. An edge weight w_{uv} is consistent with the final results of u and v when u and c are predicted to be the same label and w_{uv} is positive, and u and c are predicted to have different labels and w_{uv} is negative. Therefore, we aim to learn a weight matrix that maximizes the following function:

$$C(\mathbf{W}) = \sum_{(u,v) \in E} p_u p_v w_{uv}$$

,where C(**W**) measures the consistency of a given weight matrix.

6.1.6 Loss function

Combining the two goals, we aim to learn a weight matrix **W** via solving the following loss function:

$$min_{\mathbf{W}} Loss(\mathbf{W}) = L(\mathbf{W}) - \lambda C(\mathbf{W})$$

,where $\lambda > 0$ is a hyper-parameter to balance the two goals. We can find out that the first term is known as a simple loss function over the training dataset, while the second term is a regularization term, which we can define as consistency regularization, different from L1 normal regularization and L2 normal regularization.

6.2 GAN

GAN model traditionally contains 2 parts: a generator G, and a discriminator D. It is a game playing process with the goal

$$min_G max_D L(D, G) = E_{G_r}[logD(G_r)] + E_{z=G_g/v,L_v}[log(1 - D(G(z)))]$$

In this work, we use a variation of ordinary GAN, i.e., the conditional GAN (C-GAN). Different from ordinary GANs, it generates data according to given label. We implement it by embedding labels to features. Fig.4 shows the structure of our GAN.

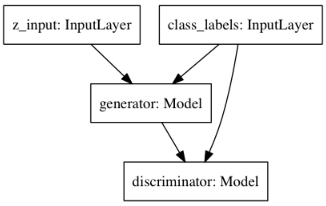


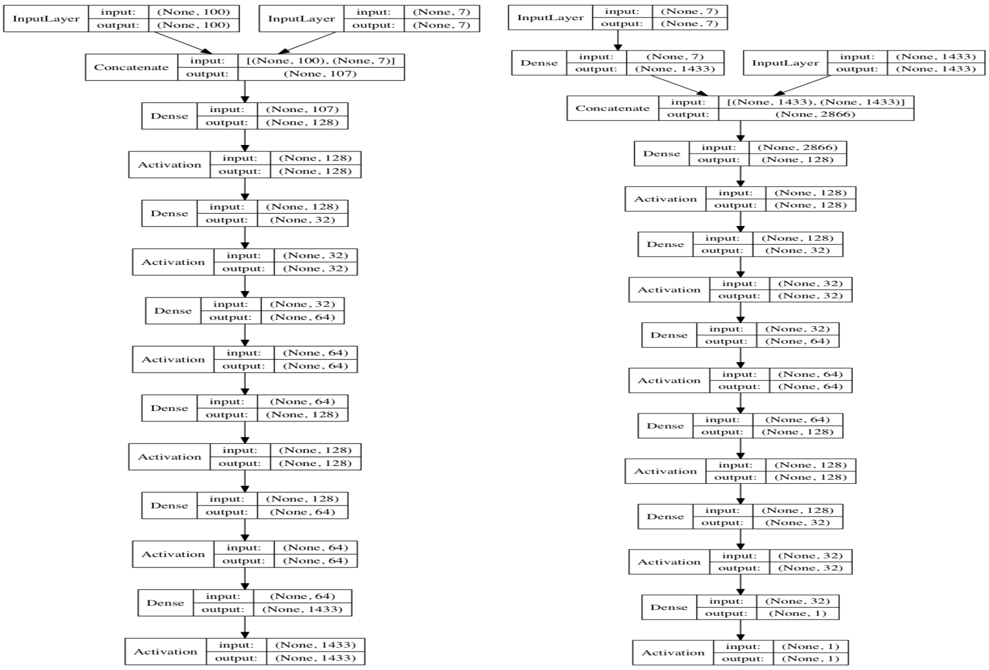
Figure 4: Generative Adversarial nets

6.2.1 Generators

We design a graph where each vertex represents a user and stores its neighborhood information (include itself) in a GNN learned vector. In this graph, labels of all the other vertices are known but vertex v. Its self representation vector is also missed. We construct one generator that generates v's self representation vector according to v's neighborhood information and each supposed label.

In other words, the generator is trying to generate $P(v|G_g/v, L_v, \theta_g)$ and encode it into graph information vector of vertex v. V's self-representation vector is then added into its neighborhood vector and forms a matrix for the whole graph. The matrix is then passing to D.

Since the network structure is quite simple, we design 6 fully connected layers in the Generator.


InputLayer input: (None, 100) output: (None, 100)
InputLayer input: (None, 7) output: (None, 7)
Concatenate input: [(None, 100), (None, 7)] output: (None, 107)
Dense input: (None, 107) output: (None, 128)
Activation input: (None, 128) output: (None, 128)
Dense input: (None, 128) output: (None, 32)
Activation input: (None, 32) output: (None, 32)
Dense input: (None, 32) output: (None, 64)
Activation input: (None, 64) output: (None, 64)
Dense input: (None, 64) output: (None, 128)
Activation input: (None, 128) output: (None, 128)
Dense input: (None, 128) output: (None, 64)
Activation input: (None, 64) output: (None, 64)
Dense input: (None, 64) output: (None, 1433)
Activation input: (None, 1433) output: (None, 1433)
InputLayer input: (None, 7) output: (None, 7)
Dense input: (None, 7) output: (None, 1433)
InputLayer input: (None, 1433) output: (None, 1433)
Concatenate input: [(None, 1433), (None, 1433)] output: (None, 2866)
Dense input: (None, 2866) output: (None, 128)
Activation input: (None, 128) output: (None, 128)
Dense input: (None, 128) output: (None, 32)
Activation input: (None, 32) output: (None, 32)
Dense input: (None, 32) output: (None, 64)
Activation input: (None, 64) output: (None, 64)
Dense input: (None, 64) output: (None, 128)
Activation input: (None, 128) output: (None, 128)
Dense input: (None, 128) output: (None, 32)
Activation input: (None, 32) output: (None, 32)
Dense input: (None, 32) output: (None, 1)
Activation input: (None, 1) output: (None, 1)

(a) Generator (b) Discriminator

Figure 5: Generator and Discriminator

6.2.2 Discriminator

The discriminator receives inputs from both G and ground-truth data. Once G passes a generated graph to D, D will calculate the KL divergence between the graph (matrix) and those of R. The value is then used to update G's generating parameters θ_g by

$$\theta_g(t+1) = \theta_g(t) - log(1 - D(G(G_g|G_g/v, L_v, \theta_g(t))|\theta_d(t), R)$$

.And update itself by

$$\theta_d(t+1) = \theta_d(t) - [log(D(G_r|\theta_d(t), R)) + log(1 - D(G(G_g|G_g/v, L_v, \theta_g(t))|\theta_d(t), R)]$$

7 Experiments and Results

We conduct comparative experiments using traditional Machine Learning models (including Decision Tree, SVM, GNB, KNN, Adaboost and Random Forest) and our new model GNN+GAN. Table I shows our result. Besides, in order to prove the improvement when inputting outputs of GNN to GAN, we also compare the performance of ordinary GAN and GNN-GAN.

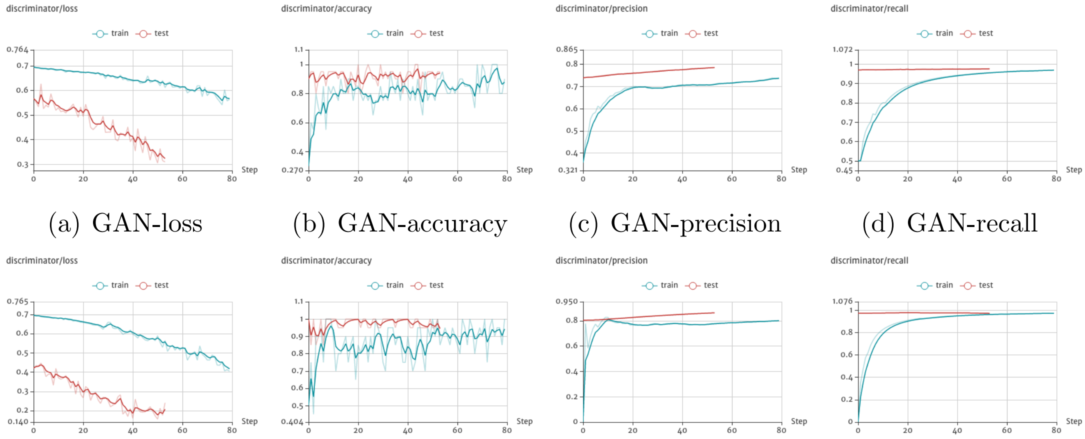
Method	Accuracy	Precision	Recall
Decision Tree	0.659	0.828	0.655
SVM	0.677	0.716	0.698
GNB	0.875	0.832	0.667
KNN	0.595	0.817	0.577
Adaboost	0.785	0.817	0.567
Random Forest	0.897	0.847	0.620
GCN(only network)	0.812	0.811	0.577
GAN	0.913	0.951	0.971
GAN+GCN	0.963	0.962	0.975

Table 1: Results of Sybil dataset

Method	Accuracy	Precision	Recall
Decision Tree	0.618	0.626	0.626
SVM	0.647	0.710	0.646
GNB	0.484	0.486	0.484
KNN	0.427	0.440	0.427
Adaboost	0.557	0.598	0.557
Random Forest	0.664	0.673	0.657
GCN	0.834	0.740	0.811
GAN	0.918	0.765	0.972
GAN+GCN	0.972	0.835	0.976

Table 2: Results of Cora dataset

Unsurprisingly, our new model beats all traditional models. This is because our model exploits more information from data. Traditional Machine Learning methods and GAN use merely features while ordinary GNN use only structure information. Our model, on the other hand, feeds information abstracted from structure to GNN, and the output predicted features from it flows into GAN for further training. Therefore, our model takes advantage both structure information and features. To make it more clear, the figure below shows the variation of discriminator's loss, accuracy, precision and recall with training epoch in each model. We leave out generator's figures because only discriminator is for future use and the curves are similar.



(a) GAN-loss

(b) GAN-accuracy

(c) GAN-precision

(d) GAN-recall

(e) GNN-GAN-loss

(f) GNN-GAN-accuracy

(g) GNN-GAN-precision

(h) GNN-GAN-recall

Figure 6: Performance of GAN and GNN-GAN

Compared to ordinary GAN, the performance of our model takes a jump when obtaining input from GNN. Besides, the converging speed also increases obviously. This verifies the information from GNN is essential.

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[2] H. Zheng, M. Xue, H. Lu, S. Hao, H. Zhu, X. Liang, and K. Ross, "Smoke screener or straight shooter: Detecting elite sybil attacks in user-review social networks," *arXiv preprint arXiv:1709.06916*, 2017.

[3] B. Wang, J. Jia, and N. Z. Gong, "Graph-based security and privacy analytics via collective classification with joint weight learning and propagation," *CoRR*, vol. abs/1812.01661, 2018.

[4] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, *et al.*, "Relational inductive biases, deep learning, and graph networks," *arXiv preprint arXiv:1806.01261*, 2018.