Article

Learning Effective Feature Representation against User Privacy Protection on Social Networks

Cheng-Te Li ^{1,*} and Zi-Yun Zeng ²¹ Institute of Data Science, National Cheng Kung University, Tainan 70101, Taiwan² Department of Statistics, National Cheng Kung University, Tainan 70101, Taiwan; ziyunn99@gmail.com

* Correspondence: chengte@mail.ncku.edu.tw; Tel.: +886-6-2757575 (ext. 53628)

Received: 7 June 2020; Accepted: 10 July 2020; Published: 14 July 2020

Abstract: Users pay increasing attention to their data privacy in online social networks, resulting in hiding personal information, such as profile attributes and social connections. While network representation learning (NRL) is widely effective in social network analysis (SNA) tasks, it is essential to learn effective node embeddings from privacy-protected sparse and incomplete network data. In this work, we present a novel NRL model to generate node embeddings that can afford data incompleteness coming from user privacy protection. We propose a structure-attribute enhanced matrix (SAEM) to alleviate data sparsity and develop a community-cluster informed NRL method, c2n2v, to further improve the quality of embedding learning. Experiments conducted across three datasets, three simulations of user privacy protection, and three downstream SNA tasks exhibit the promising performance of our NRL model SAEM+c2n2v.

Keywords: feature representation learning; node embeddings; privacy protection; social networks

1. Introduction

Network representation learning (NRL) is an essential technique in various social network analysis (SNA) and prediction tasks. The basic idea is to learn low dimensional feature representation vectors of nodes, so that the graph neighborhood of each user node can be encoded in the latent space. The performance of typical SNA tasks, such as link prediction, node classification, and community detection, can get improved due to network embeddings [1]. Several typical NRL models have been proposed, such as DeepWalk [2], LINE [3], and node2vec [4]. The common concept is to generate contexts of nodes and apply the skip-gram model [5] to learn and produce the embedding vectors. In addition to preserving network topology, some studies are devoted to incorporate node attributes when learning their embeddings. Some representative models include VGAE [6], LANE [7], and DANE [8]. They enforce nodes not only possessing similar attributes, but sharing common graph neighborhood to be as close as possible in the embedding space.

In online social networks, users are allowed to not only connect with their friends by creating new links with others, but also provide personal information, such as gender, age, interests, and employment, in their profiles. Friendships are used to create the social network while profiles are utilized as user attributes, which are associated on nodes in the network. Both social connections and user profiles are accumulated during the user-user and user-item interactions on social media [9]. Due to the effects of homophily and social circles [10], which state that users with similar attributes and common friends tend to group together, social connections and user attributes serve as essential roles when utilizing machine learning techniques to enable high-quality social services. By applying NRL methods to generate user representations, we are allowed to have better user profiling for bringing accurate applications, including computational advertising [11], identifying rumor spreaders [12], and recommender systems [13]. Although node embeddings that are derived by existing NRL models,

such as those mentioned above, have been validated to be effective, these models presume that information on the social network is complete. However, real-world social networks are usually sparse in graph topology and user attributes due to privacy concerns.

To be specific, existing NRL models need two requirements. The first is on complete network structures, meaning that users in the network have more or fewer connections. The second is on rich user profiles, referring to that most users share several personal attributes in their profiles. In other words, to the best of our knowledge, none of the existing NRL models can afford user privacy protection. User privacy protection means that users hide their personal and social information. Insufficient user information leads to incomplete specification of friendships, i.e., networks containing a number of connected components, and rare personal profiles, i.e., most of the attributes are not provided. User privacy protection is not considered into datasets used for NRL. It is unrealistic to presume that all social connections and user profiles are complete and available. As users on social media tend to be aware of privacy protection and have the right to hide personal information, incomplete graph and attribute data becomes ubiquitous. An effective NRL model is required to generate useful node embeddings that afford such data incompleteness due to user privacy protection. Privacy-aware social networking applications (e.g., friend recommendation and user profiling), powered by privacy-protected feature representation learning, will not only build the trustfulness with users, but also boost their loyalty. Nevertheless, conventional NRL models, e.g., node2vec [4] and DeepWalk [2], can still be executed to generate node embeddings from incomplete network data. However, due to lacking of rich contexts regarding edge connections and node attributes, the quality of derived node embeddings could yield unsatisfying performance in downstream tasks, such as link prediction and node classification. In other words, these NRL models require graphs with better connectivity and rich user profiles to generate effective feature representations. It is apparent that more social connections and attributes imply that users need to provide more information. Most of them could be unavailable when users concern more about privacy protection.

Few NRL models have connected user privacy with node embedding learning, such as DPNE [14] and PPGD [15]. However, their goals are to make the learned node embeddings possess the property of *differential privacy* [16], given complete social network data. That said, they are nothing to do with tackling incomplete and sparse network topology and user attributes in learning node embeddings. Another direction is to incorporate the concept of *fairness* in NRL models [17,18]. Sparse network data that result from user privacy protection are still not their concerns.

In this work, we aim at learning effective node feature representations that can afford user privacy protection, which leads to incomplete and sparse social network data. Specifically, the essential setting of our work is that users in the network may not want to provide all of their connections or fill in all of the attribute fields. The learning of each node's feature representation only relies on a limited number of edge connections to neighbors and few publicly available attributes because most of them are hidden due to the privacy concern of users. We want the learned embeddings can still capture the semantics of nodes in both structural and attributed aspects. Technically speaking, we need to mitigate the damage of the sparse graph structure and incomplete node attributes in the process of embedding learning. Assume that the effectiveness of node representations is evaluated by the performance of downstream tasks, including node classification and link prediction. We expect that the effectiveness of node representations generated by privacy-protected incomplete social network data can still approach as closely as possible to that produced by using complete graph structure and attributes to learn node embeddings. While the conventional node embedding learning methods, such as node2vec [4], DeepWalk [2], and LINE [3], do not take the data incompleteness and sparsity into consideration, the effectiveness of their generated node embeddings could be low. Our goal can also be seen as improving the effectiveness of their learned feature representations. More effectiveness improvement indicates that the derived node embeddings can better afford user privacy protection.

We will present some techniques for dealing with such privacy-incurred data incompleteness so that the learned node embeddings are still effective in downstream tasks. The main idea of our method

is two-fold. The first is the proposed Structure-Attribute Enhanced Matrix (SAEM). SAEM attempts to take full advantage of bare social connections and user attributes by making them mutually reinforced. User attributes are used to create ghost links between nodes so that the data sparsity can be alleviated. The second is the proposed community-cluster node2vec (c2n2v) model. Our c2n2v model is able to further improve the quality of NRL through incorporating topology-based user communities and attribute-based user clusters. Experiments conducted on three network datasets, along with three settings that simulate user privacy protection (i.e., random, partial, and invisible). By performing tasks of community detection, link prediction, and node classification, the results demonstrate that our model can generate better node embeddings, as comparing to some baselines and the typical node2vec method.

This work is organized, as below. We first present the problem statement in Section 3, following by elaborating the proposed method in Section 4. Subsequently, we deliver the experimental results in Section 5. We conclude this work in Section 6.

2. Related Work

Network Representation Learning (NRL). The aim of NRL is to generate node embeddings based on the graph structure and the attributes associated on nodes in the social network. The generated node representations can be applied for downstream tasks, including node classification, link prediction, and community detection. Typical NRL methods include node2vec [4], DeepWalk [2], and LINE [3]. Their generated node embeddings preserve graph neighborhood, i.e., nodes with similar structural neighbors or close to each other tend to have similar embedding vectors. Classical attribute-aware NRL methods include DANE [8], ANRL [19], and SANE [20]. Their derived node embeddings not only preserve graph proximity, but also encode similar attribute vectors of nodes. Despite the success and effectiveness of these above-mentioned NRL methods, they rely on better connectivity of graph structure and rich user attributes to learn effective node embeddings. When concerning the extremely sparse graph structure and incomplete user profiles, the quality of their generated embeddings can be less useful for downstream tasks. The goal of this paper targets at coping with the learning of node representations under such privacy-protected social network data.

Privacy-aware NRL. Some of the existing NRL methods have connected user privacy with node embedding learning, such as DPNE [14] and PPGD [15]. DPNE is a differentially private network embedding method, and PPGD is a differentially private perturbed gradient descent method for MF-based graph embedding learning. However, they target at imposing the concept of *differential privacy* [16] into the learning of node embeddings. These methods also do not concern about the data incompleteness of the input social network data. That said, the design of their technical methods is nothing to do with coping with incomplete and sparse network topology and user attributes in generating feature representations of nodes. Another relevant direction is to incorporate the concept of fairness with NRL models [17,18]. The aim is to improve the predictability of the derived node embeddings, so that the prediction performance on different demographic labels are fairly treated, i.e., not bias to the majority. Nevertheless, both the graph sparsity and attribute incompleteness that result from user privacy protection are not considered.

3. Problem Statement

We first describe the notations for our problem. Let $G = (V, E, \mathbf{A}, \mathbf{X})$ denote a social network, in which V is the node set ($n = |V|$ is the number of nodes), E is the edge set, $\mathbf{A}$ is the corresponding adjacency matrix with the symmetric form, in which element $\mathbf{A}_{ij} = 1$ if nodes $v_i, v_j \in V$ are connected (otherwise, $\mathbf{A}_{ij} = 0$), and $\mathbf{X} \in \mathbb{R}^{n \times d}$ is the node-attribute matrix, where d is the total number of attributes.

Note that the element in the attribute matrix is binary (not numerical), and all of the used benchmark datasets mentioned in Section 5 have transformed the original categorical attributes into the binary form. For example, if one attribute is gender, then it can have three dimensions depicting all

of its attribute values: “male”, “female”, and “do not apply”. The integer value d is the number of all possible categorical values of all attributes.

Next, we define the privacy exposure on each user $v \in V$. The privacy exposure, denoted by $\epsilon = [0, 1]$, is defined by the percentage of reserved information for either edge connections or node attributes. Let $\epsilon_c(v)$ be the connection privacy exposure of node v , which is defined via $\epsilon_c(v) = \frac{|\bar{E}(v)|}{|E(v)|}$, where $E(v)$ is the set of edge connections of node v , and $\bar{E}(v)$ is the set of connections that user v exposes to the public. Similarly, let $\epsilon_a(v)$ be the attribute privacy exposure of node v , which is defined via $\epsilon_a(v) = \frac{\bar{d}(v)}{d}$, where $\bar{d}(v)$ is the number of attributes that user v exposes to the public. Note that in our settings, we assume that the privacy exposure of connections and attributes for all users have the same sensitivity. It is an approximation of the realistic setting. In the experiments in Section 5, we will examine how different sensitivities on users for privacy exposure affect the effectiveness of feature representation learning.

We define the privacy-protected social network $H = (V, \bar{E}, \bar{A}, \bar{X})$ to be a subgraph of the original network G . Each user $v \in V$ has its own connection and attribute privacy exposure, $\epsilon_c(v)$, and $\epsilon_a(v)$, which result in the edge set $\bar{E} \subseteq E$ the corresponding privacy-protected adjacency matrix $\bar{A}$ and node-attribute matrix $\bar{A}$.

Privacy-Protected Network Representation Learning (PP-NRL). Given a privacy-protected social network H , PP-NRL is to learn a mapping function $f: V \rightarrow \mathbb{R}^K$ from nodes to low-dimensional embedding vectors, so that nodes sharing more similar graph neighborhood and possessing more similar attributes are as close as possible in the embedding space. Here, K is a parameter indicating the number of dimensions of feature representations, and f is a matrix of size $n \times K$, and $d \ll |V|$. While the original attribute vector is in d dimensions, the learned feature representations utilize K dimensions to encode both graph neighborhood and user attribute information. It is often that $K < d$, meaning that we can use low-dimensional dense vectors to encode information from high-dimensional sparse vectors in the graph.

Our goal is to preserve two properties from the privacy-protected network, structural proximity and semantic proximity. The former indicates that nodes with similar structural neighborhood tend to have similar embedding vectors. The latter aims at making nodes sharing similar attributes possess similar embedding vectors. We think such structural and semantic proximities are co-related with one another, and they can be simultaneously exploited to deal with the sparsity of graph topology and user attributes.

4. The Proposed Method

We first present the flowchart overview of the proposed model, as shown in Figure 1. The method consists of three phases. Given a social network, whose node attributes and edge connections are under user privacy protection (i.e., hide some information), we first perform structure-attribute mutually-reinforced representation. The goal is to create a structure-attribute enhanced matrix that mutually utilizes attributes and connections to alleviate each other's data sparsity. Subsequently, the second phase is the proposed community-cluster informed NRL. It takes SAEM as input, and enforces NRL to be aware of topology-based communities and attribute-based clusters to further improve the quality of node embeddings. The last phase is to leverage the derived node embeddings for three downstream tasks on social network analysis, including community detection, link prediction, and node classification.

Figure 1. Flowchart of the proposed Network representation learning (NRL) model.

4.1. Structure-Attribute Mutually-Reinforced Representation

The privacy-protected network H has the corresponding adjacency matrix $\bar{\mathbf{A}}$ and attribute matrix $\bar{\mathbf{X}}$. Given that both matrices $\bar{\mathbf{A}}$ and $\bar{\mathbf{X}}$ are sparse, we aim at mutually using them to reinforce the matrix representation of H , so that their sparsity can be alleviated. Eventually, a structure-attribute enhanced matrix (SAEM) will be generated. The process to construct SAEM consists of four steps.

The first step is to pre-process the adjacency matrix $\bar{\mathbf{A}}$. We perform column-wise normalization for matrix $\bar{\mathbf{A}}$ and obtain a new matrix $\hat{\mathbf{A}}$, as given by:

$$\hat{\mathbf{A}}_{ij} = \begin{cases} \frac{1}{\sum_j \bar{\mathbf{A}}_{ij}}, & (v_i, v_j) \in E \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

The second step is to compute the attribute-based similarity between nodes based on cosine score. A similarity matrix $\mathbf{S}$ can be obtained, as given by: $\mathbf{S}_{ij} = \frac{\bar{\mathbf{X}}_i \cdot \bar{\mathbf{X}}_j}{\|\bar{\mathbf{X}}_i\| \times \|\bar{\mathbf{X}}_j\|}$. Here, the range of similarity is $\mathbf{S}_{ij} \in [0, 1]$ because the transformed categorical attribute values cannot be negative, implying that the angle between two vectors cannot be greater than 90 degree. Higher $\mathbf{S}_{ij}$ indicates node v_i and v_j are more similar in terms of their attributes, because their vectors are close to each other in the space of attribute vectors. When $\mathbf{S}_{ij} = 0$, two vectors $\bar{\mathbf{X}}_i$ and $\bar{\mathbf{X}}_j$ are orthogonal, meaning that they are apart from each other (i.e., most dissimilar). Subsequently, we create some ghost links to connect node v_i and v_j based on the attribute-based similarity matrix $\mathbf{S}$ if $\mathbf{S}_{ij} \neq 0$. These ghost links will be incorporated with the normalized adjacency matrix $\hat{\mathbf{A}}$ so that the sparsity can be alleviated.

Nevertheless, matrix $\mathbf{S}$ can be too dense, because it could be not hard to share common attributes between users. Dense $\mathbf{S}$ may lead to insignificant ghost links between nodes. To this end, the third step aims to filter out dissimilar ghost links, and keep only significant ones. The idea is to only retain the k most similar nodes for every node v_i , and neglect the remaining ones. This idea is implemented by first retrieving the set of nodes that are top- k most similar to node v_i , denoted by T_i^k , where $v_i \notin T_i^k$ and $|T_i^k| = k$. Note that the top- k similarity search is implemented by locality-sensitive hashing (LSH) [21] to cope with the scalability issue. Subsequently, we use equation Equation (2) to simultaneously rule out insignificant nodes while normalizing the most similar ones, and derive a distilled attribute-based similarity matrix $\hat{\mathbf{S}}$.

$$\hat{\mathbf{S}}_{ij} = \begin{cases} \frac{\mathbf{S}_{ij}}{\sum_j \mathbf{S}_{ij}}, & v_j \in T_i^k \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

Note that k is a hyperparameter of our model, and we set $k = 5$ by default. We will examine how it affects the performance in the evaluation. Also note that we are not to add vertices into the graph. The filter of Equation (2) is also NOT to remove vertices. The purpose of the filter S_{ij} is to select the pairs of nodes that possess the higher similarity scores in terms of their attributes, and only consider these node pairs to create edges. In other words, we do not want less similar nodes to have edges in the enhanced graph because they will bring noise into the learning of node embeddings.

The last step is to generate the structure-attribute enhanced matrix (SAEM). We obtain SAEM by combining the normalized adjacency matrix $\hat{\mathbf{A}}$ with the distilled attribute-based similarity matrix $\hat{\mathbf{S}}$. The derivation of SAEM is given by:

$$\mathbf{SAEM} = (1 - \alpha) \cdot \hat{\mathbf{A}} + \alpha \cdot \hat{\mathbf{S}}, \quad (3)$$

where “+” is the operation of element-wise addition, and $\alpha \in [0, 1]$ is a hyperparameter that determines the contribution of matrices $\hat{\mathbf{A}}$ and $\hat{\mathbf{S}}$. We set $\alpha = 0.5$ by default. Higher α values rely more on attribute information while lower α values utilize more structural information. We will also look into the effect of α in the experiments. Equation (3) can be viewed as that matrix $\hat{\mathbf{S}}$ creates a number of effective ghost links, which are added into the network (i.e., $\hat{\mathbf{A}}$). Because these ghost links enhance the network connectivity, disconnected components have some potential to be merged, and the data sparsity issue can be resolved. Eventually, in the next section, the NRL part will be able to have a denser graph for learning node embeddings.

4.2. Community-Cluster Informed NRL (c2n2v)

We aim at exploiting the derived SAEM matrix to learn node embeddings. SAEM can be seen as an attribute-enhanced network, denoted by $\mathcal{H}$, with respect to the original privacy-protected network H . To be specific, let $\mathcal{H} = (V, \mathcal{E})$. If for every element $\mathbf{SAEM}_{ij} > 0$, we create an edge $e_{ij} = (v_i, v_j)$, and add it into $\mathcal{E}$, i.e., $\mathcal{E} \leftarrow \mathcal{E} \cup \{e_{ij}\}$. At this point, we can apply any typical NRL models, such as node2vec [4] and DeepWalk [2], to network $\mathcal{H}$, so as to generate node embeddings.

Given $\mathcal{H}$, in this section, we propose a community-cluster informed NRL, termed *c2n2v*, for learning node embeddings, rather than directly applying existing NRL models. We aim to make the NRL learning process aware of both structural communities and node clusters detected from network $\mathcal{H}$. The ultimate goal is to further improve the quality of node embeddings, in addition to SAEM, to overcome the issue of data sparsity. In other words, we will first detect network “communities” based on graph topology of $\mathcal{H}$, and find node “clusters” according to the node-attribute matrix $\hat{\mathbf{X}}$. Subsequently, the discovered communities and clusters are used in the process of NRL. Note that “communities” and “clusters” are different concepts. The “communities” are sets of nodes detected based on the graph structure. A community is a set of nodes that are densely connected with each other in the same set and are loosely connected with other nodes in different communities. The “clusters” are sets of nodes detected based on their attributes. A cluster is a set of nodes that share similar attributes.

Before elaborating on the details of our NRL model, we first describe some basic settings and notations. We detect communities from network $\mathcal{H}$ using Label Propagation algorithm [22], and find user clusters from matrix $\mathbf{X}$ using Affinity Propagation algorithm [23]. Note that we choose such two algorithms since they are well-known, typical, and widely effective, and they can be replaced by any state-of-the-art version. Let C and $\tilde{C}$ denote the node sets of communities and clusters, respectively. Additionally, let c and $\tilde{c}$ denote every corresponding community and cluster. In addition, our NRL model is developed based on node2vec [4].

We leverage the skip-gram architecture [5] from natural language processing to learn node embeddings in $\mathcal{H}$. In natural language processing, the skip-gram architecture learns relations between words and their contexts. Here each node in the network is treated as a word, and node sequences sampled by random walk are sampled as sentences. Let $f : V \rightarrow \mathbb{R}^K$ be the mapping function (equivalently a matrix of size $|V| \times K$ parameters) from nodes to their embeddings, where K is the

embedding dimensionality. For every node $u \in V$, we denote $N_\varphi(u) \subset V$ as the list of neighboring nodes of u by random walk sampling φ . Here the random walk [24] is a mechanism φ that, by starting from node u , randomly selects neighbor nodes in a consecutive manner, and eventually outputs a sequence of sampled nodes, denoted as $N_\varphi(u)$. Subsequently, the goal is to find the mapping function f that maximize the objective function:

$$\max_f \sum_{u \in V} \log P(N_\varphi(u)|f(u)), \quad (4)$$

which maximizes the log-probability of observing a network neighborhood $N_\varphi(u)$ conditioned on node u 's embedding vector $f(u)$.

The key step of community-cluster informed NRL lies in the generation of node sequences, i.e., the neighbor list $N_\varphi(u)$ of node u . Recall that we have the community set C and the cluster set $\tilde{C}$. Assume that the random walk sampling produces a node sequence $N_\varphi(u) = \langle n_1, n_2, \dots, n_j, \dots, n_l \rangle$, where l is the size of neighbor list (i.e., the length of random walk path). Because nodes in $N_\varphi(u)$ are what skip-gram aims to predict, to make the embedding learning be aware of either communities or clusters, we add either C or $\tilde{C}$ into $N_\varphi(u)$. In detail, take adding the communities into $N_\varphi(u)$ as an example. For $N_\varphi(u) = \langle n_1, n_2, \dots, n_j, \dots, n_l \rangle$, we have its corresponding community list $C_\varphi(u) = \langle c_1, c_2, \dots, c_j, \dots, c_l \rangle$, where $c_j \in C$. We add each $c_j \in C_\varphi(u)$ into the immediate right of its corresponding node, and obtain a new neighbor list $\hat{N}_\varphi(u) = \langle n_1, c_1, n_2, c_2, \dots, n_j, c_j, \dots, n_l, c_l \rangle$. The length of $\hat{N}_\varphi(u)$ is $2l$. We can apply the same process to add user clusters into the neighbor list. Note that since each n_i is a node and each c_j is a community, the generation of the new neighbor list $\hat{N}_\varphi(u)$ is simply to interweave n_i and c_j together into a new list. It is not a kind of concatenation operation.

In order to make the optimization problem tractable, two typical assumptions referred and justified by an existing work [4] are constructed. One is conditional independence: given the embedding of node u , the likelihood of observing u 's neighbor list $\hat{N}_\varphi(u)$ is independent of one another. Consequently, $P(\hat{N}_\varphi(u)|f(u))$ can be factorized by the neighboring nodes:

$$P(\hat{N}_\varphi(u)|f(u)) = \prod_{y \in \hat{N}_\varphi(u)} P(y|f(u)) \quad (5)$$

The second assumption is feature space symmetry: any pair of neighbors symmetrically affect each other in the space of feature representation. Therefore, for a node u , the conditional likelihood of each neighbor $y \in N_\varphi(u)$ can be implemented by a softmax function (https://en.wikipedia.org/wiki/Softmax_function) parameterized by a dot product of their feature representations:

$$P(y|f(u)) = \frac{\exp(f(y) \cdot f(u))}{\sum_{v \in V} \exp(f(v) \cdot f(u))}. \quad (6)$$

The use of exponential function in Equation (6) comes from the softmax function. The softmax function aims at taking a vector of arbitrary real numbers, and generating a probability distribution with the same number of elements such that larger elements have higher probability values. Hence, the softmax function is required to have two properties: monotonically increasing and giving a non-negative result for any real input. It turns out that the exponential function is a natural choice, since it possesses such two properties.

Eventually, the objective function can be simplified as:

$$\max_f \sum_{u \in V} \left(-\log Z_u + \sum_{y \in \hat{N}_\varphi(u)} f(y) \cdot f(u) \right), \quad (7)$$

where $Z_u = \sum_{v \in V} \exp(f(u) \cdot f(v))$ can be approximated by negative sampling [5]. In addition, this objective function can be optimized by stochastic gradient descent (SGD) [25]. Each node in the

network will be treated as the source of r random walks. These generated $r \times |V|$ random walks will be exploited to learn the embedding of each node. Unless we specify otherwise, by default, we set the length of random walks $l = 10$, and the dimension of features $K = 128$. For each node in the graph, 10 random walks ($r = 10$) are generated.

Because we can choose to include either structural community information C or user cluster information $\tilde{C}$ in embedding learning, two kinds of embedding vectors will be generated. They are community-informed embedding and cluster-informed embedding for each node $v \in V$, denoted by $f^C(v)$ and $f^{\tilde{C}}(v)$, respectively. Our goal is to let node embeddings acquire both structural communities and user clusters. Hence, such two kinds of embedding vectors are concatenated together to produce the final community-cluster informed embedding vector $\hat{f}(v)$, given by $\hat{f}(v) = [f^C(v), f^{\tilde{C}}(v)]$. Given the embedding dimensionality $K = 128$ by default, the dimensionality of $\hat{f}(v)$ is $128 \times 2 = 256$.

4.3. Downstream Tasks

Three downstream social network analysis tasks are considered in this work, including community detection, link prediction, and node classification. We choose three such downstream tasks because they are commonly used to evaluate the effectiveness of node embeddings generated by NRL methods [2–4,8,19,20]. The underlying idea is that we need to examine how effective are the feature representations of nodes. The widely-considered approach is the graph-based prediction task that accepts feature vectors as the inputs, and produces discrete labels as the outputs. Classical graph-based prediction tasks include node classification, link prediction, and community detection. They all take the feature vectors of nodes as the input of a supervised or unsupervised method, and generate labeling outcomes. An effective NRL method means that its generated node embeddings can lead to better performance of these three tasks. Below elaborates how we use the learned node embeddings to perform such three tasks. The following settings are used in our experiments.

- *Community Detection (CD)*. By regarding node embeddings as features, we apply *K-means* clustering algorithm implemented by scikit-learn (<https://scikit-learn.org/stable/>) to detect network communities. The ground-truth number of communities is given. Note that we choose K-means because it is the basic, simple, and widely-used clustering approach. Also since existing network representation learning methods [26,27] utilize K-means to fairly compare different node embedding generation models in the task of community detection, we are advised to follow accordingly.
- *Link Prediction (LP)*. This task is to predict the existence of links, given the existing network structure. We need to have the feature vectors of links and non-links. We follow node2vec [4] to employ Hadamard operator, i.e., element-wise product, to generate the feature vectors from the embeddings of two nodes. To obtain links, we remove 50% of edges chosen randomly from the privacy-protected network while ensuring that the residual network is connected. To have non-links, we randomly sample an equal number of node pairs without edges connecting them. Note that since in our experiments, we mainly target at improving node2vec under the privacy protection setting, we follow the setting of the original node2vec method [4] to randomly removing 50% of edges.
- *Node Classification (NC)*. Given a certain fraction of nodes and all of their labels, the goal is to classify the labels for the remaining nodes. Node embeddings are treated as features. We utilize one-vs-rest logistic regression classifier with L2 regularization as the classifier.

5. Experiments

The evaluation goal is to examine whether the proposed SAEM and c2n2v can lead to better performance for CD, LP, and NC tasks on various settings of user privacy protection. Besides, sensitivity analysis and visualization are conducted to show how hyperparameters influence the results and what embeddings learn.

5.1. Evaluation Settings

Datasets. We use three network datasets in the experiments, including Cora, Citeceer, and Twitter. Cora and Citeceer are benchmark network datasets (<https://lincs.soe.ucsc.edu/data>) for NRL. The Twitter dataset consists of top-10 large ego-networks (<https://snap.stanford.edu/data/ego-Twitter.html>). We present their statistics in Table 1, in which values of Twitter are averaged.

Table 1. Statistics of three network datasets.

	#Nodes	#Edges	#Components	#Attributes
Cora	2708	5429	78	1433
Citeceer	3312	4715	438	3703
Twitter	212	5054	1.6	1178

Simulation of User Privacy Protection. This evaluation is created to simulate the network incompleteness and sparsity, resulting from user privacy protection. We think the data incompleteness usually comes from that users do not want to reveal their personal information, such as friendships and attributes, due to some privacy concerns. Therefore, we use the following three strategies to generate privacy-protected social networks, and examine whether our NRL model can better survive from the sparsity. The first is **Random**: simulating every user can hide some information. We randomly remove $\beta\%$ of all edges and all users' attributes from the original social network. The second is **Partial(0.5)**: simulating only some users hide some information. We randomly remove $\beta\%$ of edges and attributes for every of 50% randomly selected users. The third is **Invisible**: simulating some users totally hide their information. We randomly remove all edges and attributes for every of $\beta\%$ randomly selected users. We repeat the $\beta\%$ randomness up to 100 times, and report the average results.

Evaluation Plans. The evaluation consists of three main parts for community detection (CD), link prediction (LP), and node classification (NC). For each part, we present the results (y-axis) on three different simulations of privacy protection by changing $\beta\%$, i.e., reserving $1 - \beta\%$ information (x-axis). Note that we do not report the results of NC for Twitter data, since it contains no node labels. In addition, we also exhibit the sensitivity of hyperparameter k and α that determines SAEM using NC. We use t-SNE [28] to visualize the learned node embeddings.

Competing Methods. (1) **adj**: using only the adjacency matrix for node features, (2) **att**: using only user attributes as node features, (3) **adj+att**: concatenating adjacency vectors and user attributes as node features, (4) **n2v**: the typical NRL model node2vec [4], (5) **SAEM**: using the proposed enhanced matrix for node features, (6) **SAEM+n2v**: applying node2vec to SAEM for embedding learning, and (7) **SAEM+c2n2v**: combining the proposed SAEM with our c2n2v to generate node embeddings. Note that for link prediction, we do not have the first three competing methods because Hadamard product is used for only embedding vectors.

Evaluation Metrics. For community detection, we employ Normalized Mutual Information (NMI) [29]. For link prediction, we utilize Area Under Curve (AUC) scores. For node classification, we consider Macro-F1 (MAF) as the metric. Higher values indicate better performance in all metrics. We consider these evaluation metrics because they are benchmarks and typically and widely used for experiments in the literature of network representation learning [2–4,8,19,20].

5.2. Experimental Results

Community Detection. The results are exhibited in Figure 2. We can find that the proposed SAEM+c2n2v outperforms the other competing methods on three privacy protection cases across three datasets. Although SAEM by itself cannot work well, it is able to significantly boost the performance of n2v. In addition, when users are simulated to protect more privacy, i.e., revealing less information in the x-axis, the superiority of our SAEM-based NRL models tends to be more significant, as comparing to n2v. Such results imply that SAEM is capable of effectively combining graph structure and user

attributes to alleviate the issue of data incompleteness, and eventually generating high-quality node embeddings. Moreover, it is interesting to see that the superiority of SAEM+cn2nv on Twitter data are not so significant as the other datasets. We think the possible reason is that Twitter data is not so sparse, i.e., the number of disconnected components is quite small, as presented in Table 1.

Link Prediction. We show the results in Figure 3. It can be found that the tendency of LP task is quite similar to that of the CD task. SAEM+cn2nv generates the highest AUC scores in all cases. There is one additional finding. The superiority gap between our SAEM-based NRL models and other methods is larger than those in CD task on Cora and Citeseer, and the superiority is smaller on Twitter. Here is the potential reason. Link prediction needs to generate features of node pairs. Since disconnected components lead to worse-quality node embeddings, features of node pairs based on Hadamard operator are accordingly ineffective in depicting the correlatedness between nodes. These results deliver an insight: SAEM particularly works on social networks with more disconnected components, and it is especially useful for friendship recommendation.

Figure 2. Results of Normalized Mutual Information (NMI) scores on community detection across three privacy settings and three datasets.

Figure 3. Results of Area Under Curve (AUC) scores on link prediction for three privacy settings and three datasets.

Node Classification. The results are exhibited in Figure 4. Our SAEM-based NRL models again outperform the other competing methods. SAEM itself can be seen as an effective feature representation, especially when users protect more privacy (i.e., lower “reserved (%)”). Nevertheless, the superiority gaps are not so significant as those on CD and LP tasks. The classifier is able to some extent distinguish different labels, so the negative effect of disconnected components is not strong.

n2v vs. c2n2v. To further understand the effectiveness of the proposed c2n2v, we provide a detailed comparison between n2v and c2n2v with SAEM, as shown in Table 2. The results show that c2n2v can moderately improve the performance across datasets, tasks, and privacy settings.

t-SNE Visualization and Explainability. The model explainability and interpretability is an essential issue in deep learning and artificial intelligence models [30–33]. People usually expect that a black-box model can deliver some intuitive concept while producing satisfying performance. Here, we would like to present the model explainability of the proposed privacy-aware NRL via visualization. We exhibit the results of t-SNE visualization for the typical NRL model node2vec (n2v) and our SAEM+c2n2v in Figure 5. The t-SNE visualization can be also considering as explaining the effect of the proposed SAEM method. The interpretability comes from how nodes with different color labels distribute in the t-SNE embedding space. It can be apparently found that our model is able to effectively group nodes with the same labels (colors) together. While disconnected components in the original privacy-protected network worsen the quality of embeddings, our model can successfully alleviate the sparsity problem and generate better node embeddings. Because the learned node embeddings can be essentially treated as node feature vectors in a latent form, such visualization results can explain that the proposed SAEM can better transform the original social network data to effective feature representations, in which nodes with the same labels are close to one another.

Hyperparameter Analysis. By varying k and α that affect the effectiveness of SAEM, we show the results of hyperparameter sensitivity in Figure 6. There are two findings. First, the outcome informs us that under any fixed k , the value of α does not affect the performance of NC tasks. Second, choosing the most relevant nodes based on attribute-based similarity by k in Equation (2) has a crucial impact on the performance. The results indicate that selecting only a few highly similar nodes (i.e., smaller k) is sufficient to create useful ghost links when constructing SAEM. In practice, we would suggest the data scientists to choose smaller k , which can select nodes with the most similar attributes to mitigate the sparsity of graph structure. Larger k may make irrelevant nodes to be connected with one another. Besides, as for choosing the value of α that is in charge of enhancing the data incompleteness, we think the SAEM should blend graph structure and node attributes together. Hence, we would suggest the data scientists to choose α to be around 0.5 to avoid the risk of extremely sparse graph structure and drastically incomplete user attributes.

Table 2. Comparison between n2v and c2n2v with structure-attribute enhanced matrix (SAEM).

		Cora			Citeseer		
		Random	Partial (0.5)	Invisible	Random	Partial (0.5)	Invisible
CD	n2v	0.422	0.422	0.397	0.391	0.384	0.390
	c2n2v	0.434	0.432	0.407	0.396	0.388	0.395
LP	n2v	0.846	0.850	0.848	0.913	0.912	0.914
	c2n2v	0.854	0.858	0.857	0.918	0.918	0.920
NC	n2v	0.706	0.709	0.704	0.635	0.631	0.633
	c2n2v	0.711	0.710	0.714	0.640	0.638	0.638

Figure 4. Results of Macro-F1 (MAF) scores on node classification for three privacy settings and three datasets.

Figure 5. t-SNE visualization between n2v and SAEM+c2n2v in Cora data. Colors indicate labels of nodes.

Figure 6. Sensitivity analysis for hyperparameters k and α that determines SAEM, plotted using NC task in Cora data.

6. Conclusions

To the best of our knowledge, this work is the first attempt to learn effective node embeddings that can afford user privacy protection on social networks. We summarize the contributions of

this work. First, we propose the structure-attribute enhanced matrix (SAEM) to alleviate the data sparsity produced by privacy protection. Second, the proposed c2n2v can further improve the quality of node embeddings by incorporating both structural communities and user clusters. Third, experiments conducted on three datasets, three tasks, and three different simulations of user privacy protection show the promising performance of SAEM+c2n2v. Our future work is aimed at applying and generalizing SAEM to other NRL models, and to intelligently build SAEM via reinforcement learning.

Author Contributions: Conceptualization, C.-T.L.; Data curation, Z.-Y.Z.; Formal analysis, Z.-Y.Z.; Funding acquisition, C.-T.L.; Investigation, C.-T.L. and Z.-Y.Z.; Methodology, C.-T.L. and Z.-Y.Z.; Project administration, C.-T.L.; Resources, C.-T.L.; Software, Z.-Y.Z.; Supervision, C.-T.L.; Validation, Z.-Y.Z.; Visualization, Z.-Y.Z.; Writing—original draft, C.-T.L.; Writing—review & editing, Z.-Y.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the Ministry of Science and Technology (MOST) of Taiwan under grants 109-2636-E-006-017 (MOST Young Scholar Fellowship) and 108-2218-E-006-036, and also by Academia Sinica under grant AS-TP-107-M05.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Cui, P.; Wang, X.; Pei, J.; Zhu, W. A Survey on Network Embedding. *IEEE Trans. Knowl. Data Eng. TKDE* **2018**, *31*, 833–852. [[CrossRef](#)]
2. Perozzi, B.; Al-Rfou, R.; Skiena, S. DeepWalk: Online Learning of Social Representations. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD, New York, NY, USA, 24–27 August 2014; pp. 701–710.
3. Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; Mei, Q. Line: Large-scale information network embedding. In Proceedings of the 24th International Conference on World Wide Web, WWW, Florence, Italy, 18–22 May 2015; pp. 1067–1077.
4. Grover, A.; Leskovec, J. node2vec: Scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD, San Francisco, CA, USA, 13–17 August 2016; pp. 855–864.
5. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. *arXiv* **2013**, arXiv:1301.3781.
6. Kipf, T.N.; Welling, M. Variational graph auto-encoders. *arXiv* **2016**, arXiv:1611.07308.
7. Huang, X.; Li, J.; Hu, X. Label informed attributed network embedding. In Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM, Cambridge, UK, 6–10 February 2017; pp. 731–739.
8. Gao, H.; Huang, H. Deep Attributed Network Embedding. In Proceedings of International Joint Conferences on Artificial Intelligence, IJCAI, Stockholm, Sweden, 13–19 July, 2018; pp. 3364–3370.
9. Benevenuto, F.; Rodrigues, T.; Cha, M.; Almeida, V. Characterizing User Behavior in Online Social Networks. In Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement, IMC '09, Chicago IL, USA, 4–10 November 2009; pp. 49–62.
10. McAuley, J.; Leskovec, J. Learning to Discover Social Circles in Ego Networks. In Proceedings of the 25th International Conference on Neural Information Processing Systems, NIPS '12, Lake Tahoe, NV, USA, 3–6 December 2012; Volume 1, pp. 539–547.
11. Liu, H.; Pardoe, D.; Liu, K.; Thakur, M.; Cao, F.; Li, C. Audience Expansion for Online Social Network Advertising. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, San Francisco, CA, USA, 13–17 August 2016; pp. 165–174.
12. Rath, B.; Gao, W.; Ma, J.; Srivastava, J. Utilizing computational trust to identify rumor spreaders on Twitter. *Soc. Netw. Anal. Min.* **2018**, *8*, 64:1–16. [[CrossRef](#)]
13. Guy, I. Social Recommender Systems. In *Recommender Systems Handbook*; Springer: Boston, MA, USA; **2015**; pp. 511–543.
14. Xu, D.; Yuan, S.; Wu, X.; Phan, H. DPNE: Differentially Private Network Embedding. In Proceedings of Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD, Melbourne, Australia, 3–6 June 2018; pp. 235–246.

15. Zhang, S.; Ni, W. Graph Embedding Matrix Sharing With Differential Privacy. *IEEE Access* **2019**, *7*, 89390–89399. [[CrossRef](#)]
16. Dwork, C. Differential Privacy. In Proceedings of the 33rd International Colloquium on Automata, Languages and Programming, part II (ICALP 2006, Venice, Italy, 10–14, July 2006); Springer: Berlin/Heidelberg, Germany, 2006; Volume 4052, *Lecture Notes in Computer Science*, pp. 1–12.
17. Rahman, T.; Surma, B.; Backes, M.; Zhang, Y. Fairwalk: Towards Fair Graph Embedding. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI, Macao, China, 10–16 August 2019; pp. 3289–3295.
18. Bose, A.; Hamilton, W. Compositional Fairness Constraints for Graph Embeddings. In Proceedings of the 36th International Conference on Machine Learning, ICML, Long Beach, CA, USA, 9–15 June 2019; pp. 715–724.
19. Zhang, Z.; Yang, H.; Bu, J.; Zhou, S.; Yu, P.; Zhang, J.; Ester, M.; Wang, C. ANRL: Attributed Network Representation Learning via Deep Neural Networks. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18. International Joint Conferences on Artificial Intelligence Organization, Stockholm, Sweden, 13–19 July 2018; pp. 3155–3161.
20. Wang, H.; Chen, E.; Liu, Q.; Xu, T.; Du, D.; Su, W.; Zhang, X. A United Approach to Learning Sparse Attributed Network Embedding. In Proceedings of the IEEE International Conference on Data Mining, ICDM, Singapore, Singapore, 17–20 November 2018; pp. 557–566.
21. Wang, J.; Shen, H.T.; Song, J.; Ji, J. Hashing for Similarity Search: A Survey. *arXiv* **2014**, arXiv:cs.DS/1408.2927.
22. Raghavan, U.N.; Albert, R.; Kumara, S. Near linear time algorithm to detect community structures in large-scale networks. *Phys. Rev. E* **2007**, *76*, 036106. [[CrossRef](#)] [[PubMed](#)]
23. Frey, B.J.; Dueck, D. Clustering by passing messages between data points. *Science* **2007**, *315*, 972–976. [[CrossRef](#)] [[PubMed](#)]
24. Lovasz, L. Random Walks on Graphs: A Survey. *Bolyai Soc. Math. Stud.* **1993**, *2*, 1–46.
25. Bottou, L. Stochastic gradient learning in neural networks. *Proc. Neuro-Nimes* **1991**, *91*.
26. Tian, F.; Gao, B.; Cui, Q.; Chen, E.; Liu, T.Y. Learning Deep Representations for Graph Clustering. In Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, AAAI '14, Québec City, QU, Canada, 27–31 July 2014; pp. 1293–1299.
27. Chen, Z.; Chen, C.; Zhang, Z.; Zheng, Z.; Zou, Q. Variational Graph Embedding and Clustering with Laplacian Eigenmaps. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19, Macao, China, 10–16 August 2019, 2019; pp. 2144–2150.
28. van der Maaten, L.; Hinton, G. Visualizing Data using t-SNE. *J. Mach. Learn. Res.* **2008**, *9*, 2579–2605.
29. Yang, J.; Leskovec, J. Defining and Evaluating Network Communities Based on Ground-Truth. In Proceedings of the 2012 IEEE 12th International Conference on Data Mining, ICDM '12, Brussels, Belgium, 10 December 2012; pp. 745–754.
30. Adadi, A.; Berrada, M. Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access* **2018**, *6*, 52138–52160. [[CrossRef](#)]
31. Kuhl, N.; Lobana, J.; Meske, C. Do you comply with AI? – Personalized explanations of learning algorithms and their impact on employees' compliance behavior. *arXiv* **2020**, arXiv:cs.CY/2002.08777.
32. Meske, C.; Bunde, E. Using Explainable Artificial Intelligence to Increase Trust in Computer Vision. *arXiv* **2020**, arXiv:cs.HC/2002.01543.
33. Murdoch, W.J.; Singh, C.; Kumbier, K.; Abbasi-Asl, R.; Yu, B. Definitions, methods, and applications in interpretable machine learning. *Proc. Natl. Acad. Sci. USA* **2019**, *116*, 22071–22080. [[CrossRef](#)] [[PubMed](#)]