Applied Network Science

RESEARCH Open Access

A flexible framework for multiple-role discovery in real networks

Shu Liu[1*], Fujio Toriumi[1], Mao Nishiguchi[1] and Shohei Usui[2]

*Correspondence: liu@torilab.net

[1] School of Engineering, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8654, Japan
[2] DSOC (Data Strategy & Operation Center), Sansan Inc., Tokyo, Japan

Abstract
In complex networks, the role of a node is based on the aggregation of structural features and functions. However, in real networks, it has been observed that a single node can have multiple roles. Here, the roles of a node can be defined in a case-by-case manner, depending on the graph data mining task. Consequently, a significant obstacle to achieving multiple-role discovery in real networks is finding the best way to select datasets for pre-labeling. To meet this challenge, this study proposes a flexible framework that extends a single-role discovery method by using domain adversarial learning to discover multiple roles for nodes. Furthermore, we propose a method to assign sub-networks, derived through community extraction methods, to a source network and a validation network as training datasets. Experiments to evaluate accuracy conducted on real networks demonstrate that the proposed method can achieve higher accuracy and more stable results.

Keywords: Complex network, Role discovery, Multi-label classification, Real network, Community extraction, Adversarial learning

Introduction

Complex networks have a wide range of applications in biology, social science, and other fields. The role of a node involves the aggregation of structural features or functions in complex networks. In network topologies, the roles of nodes include star-center, star-periphery, and clique-member (Rossi and Ahmed 2014). On the other hand, in directed three-node motifs, according to the direction of the link, a node can be simply assigned to the role of source or sink. Role discovery is the task of assigning role labels to nodes (Rossi and Ahmed 2014). Discovering the roles of nodes involves creating role labels on nodes, providing a more intuitive understanding of network structures, and elucidating the mechanisms and hidden higher-order knowledge of large-scale complex systems. Role discovery is vital for graph mining and exploratory analysis, and it is helpful in many practical applications. For example, in the detection of anomalies in technical networks such as IP traces (Mahadevan et al. 2006; Rossi et al. 2013a), nodes that do not fit the normal role (i.e., the usual structural pattern) can be identified and removed by assigning them the role of anomaly (Rossi et al. 2013b). In online social networks, users having the role of "star-center" can be regarded as influencers, and they are considered to be highly effective in delivering advertisements.

A single node may have multiple roles simultaneously in real-world networks, depending on the network construction strategy. In temporal networks, a node's structure (or function) may change over time, which implies that its roles may thus change over time. Depending on the granularity of partitioning, multiple roles can be observed within a single time slice (Rossi and Ahmed 2014). For example, in a human-relationship network in which people are nodes, and the connections between those people are links, a node that plays the role of a team leader in a company during weekdays forms a local structure with a tree topology (reflecting the structure of the company). In contrast, that node could also have the role of clique-member on weekends in a family where the members may have dense interactions. In addition, in a lexical co-occurrence network, if the part of speech of a lexeme is used as a role label, a lexeme with multiple parts of speech will have multiple roles. The phenomenon of multiple roles of nodes exists in a broad range of domains, not exclusively in temporal networks. Essentially, this phenomenon is determined by the local structures of the network. Suppose one node appears in multiple local structures and plays different roles. In that case, multiple role phenomena can be observed on a global scale.

Although previous research has acknowledged the multiple-role phenomenon, few specific approaches to the estimation of multiple roles have been presented (Rossi and Ahmed 2014; Liu et al. 2021). In earlier work (Liu et al. 2021), we proposed a method to predict multiple roles with an adversarial learning approach by treating the multiple-role discovery task as a multi-label classification problem. Specifically, that method applies struc2vec (Ribeiro et al. 2017) to gain the structural embedding features of nodes from network structure and then feeds the embedding features to a domain adversarial learning framework to predict the multiple role labels. In this study, we aim to extend our previous method (Liu et al. 2021) to develop a versatile framework that can predict multiple roles within the scenario of real networks. First, to confirm the flexibility of our previously proposed method (Liu et al. 2021), we replaced struc2vec with two other popular structural embedding methods, GraphWave (Donnat et al. 2018) and role2vec (Ahmed et al. 2020). Second, we further tested the effectiveness of selecting the source networks and validation networks through the community extraction method as an alternative to random sampling. In this paper, we apply the proposed method to real-world networks (Wikipedia network and Blogcatalog network (Zhang et al. 2019)) to validate its accuracy.

Related work

Role discovery

Role discovery is the task of clustering nodes with a predefined equivalence into classes with different roles. Here, equivalence can be divided into two major types: structural equivalence, which is defined by the network structure, and feature equivalence, which is extracted from the features (or functions) of the nodes. Roles can be divided into graph-based roles and feature-based roles based on the definition of each major equivalence. Methods aimed at role discovery obtain a vector representation of nodes from the network structure, cluster nodes by specific similarity criteria, and assign role labels to each cluster. In the task of role assignment, the conventional methods can be divided into three types: hard single-role discovery (assigning each node to one specific role), soft single-role discovery (assigning each node to one of a distribution of roles), and multiple-role discovery (assigning each node to one or more roles) (Rossi and Ahmed 2014). A large number of works have attempted hard single-role discovery. The most famous approach is the block model (Arabie et al. 1978; Holland et al. 1983; Nowicki and Snijders 2001; Batagelj et al. 2004). The block model is typically formulated as an optimization problem using a well-formed objective function, where nodes with the same role are aggregated and represented as blocks (nodes), and the edges between the blocks construct a role-interaction graph showing the interactions between the roles. Some models assign each node to one of several roles (or blocks). Other methods use a form of similarity between the rows of the adjacency matrix to calculate the roles (Burt 1976; Brandes and Lerner 2010). The main flow of these methods is to first calculate the similarity (or distance) between each pair of rows of the adjacency matrix and then use the resulting similarity matrix to cluster the nodes. In addition, spectral methods use a subset of the eigenvectors of the adjacency matrix (or similarity matrix) to derive the roles (Brandes and Lerner 2010). These methods are unsupervised learning approaches, and thus they have the disadvantage of requiring subjective guesswork when assigning specific roles to individual clusters, making the roles less interpretable and the validity of the assigned roles more difficult to check. On the other hand, the supervised learning approach trains the learning model through predefined roles, role assignment criteria, and positive examples to predict the roles of the test data. Since the roles are defined as needed, interpretability is enhanced and the accuracy of the predicted results can be maintained at a certain level by learning from positive examples. Our previous work on node role discovery using transfer learning (Kikuta et al. 2020) is an example of a supervised learning approach, in which role knowledge from a pre-labeled network (source network) is transferred to a network whose roles are unknown (target network). In addition, previous research has taken the approach of using a network that validates the model's accuracy (validation network) to select the optimal hyperparameters by grid search.

In contrast, as a soft single-role discovery method, Airoldi et al. (2008) proposed a mixed membership stochastic block model that relaxes the assumption of nodes belonging to only a single role. This model assigns each node a probability vector (where the sum of the vector elements equals 1) belonging to multiple roles instead of assigning it a single role label. Accordingly, this method is inadequate for multiple-role discovery because it can only provide a rough trend of the node's multiple roles rather than the specific multiple roles.

To our best knowledge, we proposed the first method for multiple-role discovery (Liu et al. 2021), which was derived from the observation that nodes may have multiple roles simultaneously in real networks. We applied the multi-label classification used in neural networks to a previously proposed hard single-role discovery framework (STV method) (Kikuta et al. 2020).

Roles in real networks

In the context of network topology, the role of a node is defined as star-center, star-periphery, or clique member. As mentioned above, the role is based on an aggregation of the structural features or functions of the nodes. In different contexts, roles have different interpretations. From the perspective of information mining, it is necessary to define more suitable role types for specific real networks. For example, in a lexical network where words are nodes and co-occurrence phenomena are represented by edges, the structural features of words can be regarded as an aggregation of grammatical functions. One of the concepts defined by the grammatical criteria of a word is its part of speech (Baker and Croft 2017), so part of speech can be used as the word's role in a lexical network. However, a word can have more than one part-of-speech label, so predicting the word's part of speech can be considered a task of multiple-role discovery.

Note that in information mining for real networks, the structural features always appear in functional labels. For instance, if a star graph is considered a local structure in human relationship networks, the person with the star-center role (in the structural sense) can be treated as a leader or influencer (in the functional sense). Therefore, it is crucial to pre-define the practical label set when given a target real network. Furthermore, it is evident that the suitable label set in one real network might differ from those in other real networks. Consequently, it is essential to select the appropriate training data (source network). When considering the problem of how to create training data (source network), a straightforward approach is to label randomly sampled data instances from the entire data. However, the following problems may arise regarding the roles of nodes in complex networks.

1 Due to the scale-free nature of complex networks, there is a phenomenon where a small number of nodes have a high degree. It is doubtful that random sampling can obtain nodes with such characteristics, which leads to missing important roles in the training data.
2 Since role labeling is often based on the roles of surrounding nodes, it is necessary to check not only the randomly sampled nodes but also the nodes surrounding them up to several hops away. Due to the small-world nature of complex networks, it may be necessary to consider the entire network. To this end, the roles of most nodes in the entire network have to be labeled to build the training data, making random sampling meaningless.

Thus, building training datasets for role discovery in real networks is challenging because (1) the roles themselves are diverse and need to be defined according to the meaning of the network and (2) the scale-free and small-world nature of complex networks make random sampling impractical.

Embedding methods for structural features

Learning node embedding is a crucial task in obtaining the representation vector for each node. Formally, given a network $G = (V, E)$, the goal is to learn a function $f : V \to \mathbb{R}^{|V| \times k}$, where $k \ll |V|$, so that similar nodes (e.g., nodes near each other or with similar functions) lie close together in the low k-dimensional space. This makes it possible to apply vector-based machine learning methods in graph mining, such as link prediction (Lü and Zhou 2011; Kumar et al. 2020), node and link classification (Bhagat et al. 2011; Tang et al. 2016), and anomaly detection (Akoglu et al. 2015).

Many works have attempted learning embedding (Goyal and Ferrara 2018; Cai et al. 2018). Rossi et al. (2020) categorized those works into community-based and role-based embedding methods. The former type is based on proximity similarity, which embeds nodes near neighbors or nodes in the same community. DeepWalk (Perozzi et al. 2014) is a well-known method that applies random walk to generate sequences of nodes, employing a Skip-Gram model to create node embeddings. node2vec (Grover and Leskovec 2016) further extended DeepWalk with a bias to leverage nodes' homophily and structural equivalency. LINE (Tang et al. 2015) focuses on first- and second-order neighbors to learn the representation vectors. The latter type, role-based embedding methods, is based on structural equivalence, which embeds nodes near each other if they share similar connection patterns, even though these nodes may be far away in the network. On the other hand, struc2vec (Ribeiro et al. 2017) calculates the similarities of the distribution for the neighbor's degree in multiple scales, applies biased random walk, and generates embeddings via the Skip-Gram approach. role2vec (Ahmed et al. 2020) extracts structural features (e.g., statistics of graphlets for all nodes), clusters nodes into role groups, applies random walk to generate sequences of roles for the node, learns the embeddings of roles via the Skip-Gram approach, and treats the roles' embedding as the nodes' embedding. GraphWave (Donnat et al. 2018) treats wavelets as probability distributions and uses empirical characteristic functions of the distribution to form the structural embeddings of nodes.

Recent work (Jin et al. 2022) on evaluating the structural node embeddings maintains that different variables, such as choice of similarity measure, classifier, and label definitions, lead to different results for the downstream graph mining tasks (e.g., node classification). This paper focuses on testing the proposed methods' generalization capability and flexibility on different embeddings instead of choosing the winner of the embedding methods. Moreover, note that the definitions of the node label used in this research are not equivalent to the structure due to the lack of benchmark datasets.

Proposed framework

Notions

Given a network $G = (V, E)$, where $V = v_1, \ldots, v_n$ represents the node set, $E = e_1, \ldots, e_m$ denotes the edge set. The source network G_s and validation network G_v are separated from G by the community extraction method, and the remaining network $G \backslash \{G_s, G_v\}$ forms the target network G_t.

$f_e : G_s, G_t, G_v \to \mathbb{R}^{(|V_s|+|V_t|+|V_v|)\times k}$ denotes the embedding method that extracts the structural features $\theta_s = \mathbb{R}^{|V_s|\times k}$, $\theta_t = \mathbb{R}^{|V_t|\times k}$, and $\theta_v = \mathbb{R}^{|V_v|\times k}$ for each node in G_s, G_t, and G_v, respectively.

Let $\mathcal{L} = l_1, \ldots, l_j$ be the role label set and $\mathcal{P}(\mathcal{L})$ represent the powerset of $\mathcal{L}$ for any possible label combination $L \in \mathcal{P}(\mathcal{L})$. For single-role discovery tasks, the purpose is to find the optimal function $f_{srd} : \theta_t \to \{l_1, \ldots, l_j\}^{|V_t|}$. θ_t is the target network's structural features learned by the embedding method, and $\{l_1, \ldots, l_j\}^{|V_t|}$ denotes the role vector of the target network, in which the elements denote the related single role. In contrast, the goal of multiple-role discovery tasks is to learn the optimal function $f_{mrd} : \theta_t \to L^{|V_t|}$. $L \in \mathcal{P}(\mathcal{L})$ denotes the multiple roles for the related node.

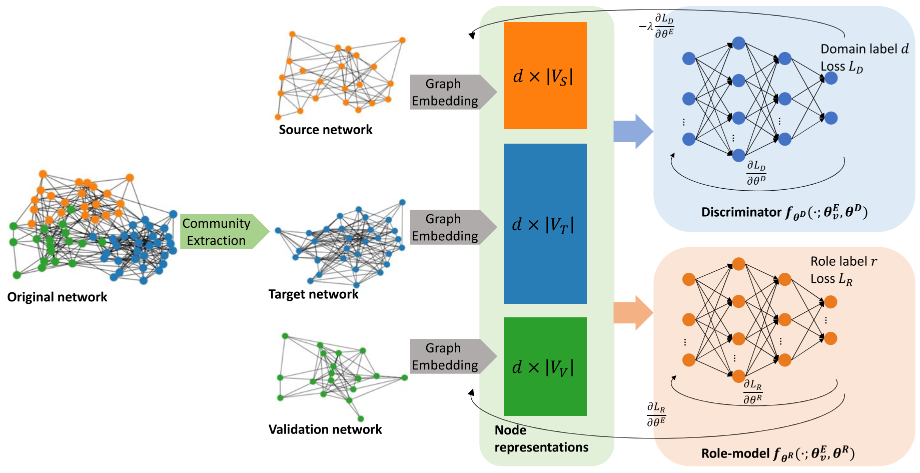

Source network
Original network
Community Extraction
Target network
Validation network
Graph Embedding
Graph Embedding
Graph Embedding
$d \times |V_S|$
$d \times |V_T|$
$d \times |V_V|$
Node representations
$-\lambda \frac{\partial L_D}{\partial \theta^E}$
Domain label d
Loss L_D
$\frac{\partial L_D}{\partial \theta^D}$
Discriminator $f_{\theta^D}(\cdot;\theta_v^E,\theta^D)$
Role label r
Loss L_R
$\frac{\partial L_R}{\partial \theta^R}$
$\frac{\partial L_R}{\partial \theta^E}$
Role-model $f_{\theta^R}(\cdot;\theta_v^E,\theta^R)$

Fig. 1 Overview of proposed framework

Framework

An overview of the proposed framework for multiple-role discovery in real networks is shown in Figure 1. Since we previously proposed the fundamental idea of the framework (Liu et al. 2021), in this paper, we integrate and fortify it, test the validity of the framework, and evaluate the replaceability of the components in the framework. Algorithm 1 shows the pseudo-code for the proposed framework.

Algorithm 1 Proposed framework

```
Require: network $G = (V, E)$, role set $\mathcal{L}$, role labeling rules $Rule$, embedding size $k$, hidden layer size
    $H$, learning mode $M \in \{BPMLL, BPMLL\_SD2, BPMLL\_SD3, BPMLL\_DD\}$
Ensure: multiple roles $L^{|V|}$, where $L \in \mathcal{P}(\mathcal{L})$
 1: Split $G$ to $G_s = (V_s, E_s)$, $G_v = (V_v, E_v)$, $G_t = (V_t, E_t)$ by community extraction method
 2: Obtain $\theta_s = \mathbb{R}^{|V_s| \times k}$, $\theta_v = \mathbb{R}^{|V_v| \times k}$, $\theta_t = \mathbb{R}^{|V_t| \times k}$ of $G_s$, $G_v$, $G_t$ through graph embedding
    method
 3: Label $L^{|V_s|}$, $L^{|V_v|}$ for $G_s$, $G_v$ by $Rule$
 4: if $M == BPMLL$ then
 5:     $\theta^R \leftarrow random\_init(H)$
 6:     repeat
 7:         #Update $\theta^R$ by gradient descent according to Equation 2
 8:         $E_R(V_s;\theta^R) = -\sum_{i=1}^{|V_s|} \frac{1}{|L_i||\bar{L}_i|} \sum_{(k,l)\in L_i \times \bar{L}_i} \exp(-(f_{mrd}^{\theta^R}(\theta_s)_k^i - f_{mrd}^{\theta^R}(\theta_s)_l^i))$
 9:         $\theta^R \leftarrow \theta^R - \mu_R \frac{\partial E_R}{\partial \theta^R}$
10:     until Converge
11: else
12:     $\theta^R \leftarrow random\_init(H)$
13:     $\theta^D \leftarrow random\_init(H)$
14:     repeat
15:         #Update $\theta^D$ by gradient descent
16:         $E_D = J(V;\theta^D, \theta_s, \theta_v, \theta_t)$
17:         $\theta^D \leftarrow \theta^D - \mu_D \frac{\partial E_D}{\partial \theta^D}$
18:         #Update $\theta^R, \theta_s, \theta_v, \theta_t$ by gradient descent according to Equation 2
19:         $E_R(V_s;\theta^R, \theta_s) = -\sum_{i=1}^{|V_s|} \frac{1}{|L_i||\bar{L}_i|} \sum_{(k,l)\in L_i \times \bar{L}_i} \exp(-(f_{mrd}^{\theta^R}(\theta_s)_k^i - f_{mrd}^{\theta^R}(\theta_s)_l^i))$
20:         $\theta^R \leftarrow \theta^R - \mu_R \frac{\partial E_R}{\partial \theta^R}$
21:         $\theta_s \leftarrow \theta_s - \mu_R(\frac{\partial E_R}{\partial \theta_s} - \lambda \frac{\partial E_D}{\partial \theta_s})$, $\theta_v \leftarrow \theta_v - \mu_R(\frac{\partial E_R}{\partial \theta_v} - \lambda \frac{\partial E_D}{\partial \theta_v})$, $\theta_t \leftarrow \theta_t - \mu_R(\frac{\partial E_R}{\partial \theta_t} - \lambda \frac{\partial E_D}{\partial \theta_t})$
22:     until Converge
23: end if
24: $L^{|V_t|} \leftarrow f_{mrd}^{\theta^R}(\theta_t)$
```

Dataset building

Considering the challenges in building training datasets for role discovery in real networks (mentioned in section), as shown in line 1 of Algorithm 1, we propose creating datasets by community extraction methods, such as the Louvain method (Blondel et al. 2008) and the Leiden method (Traag et al. 2019). Community extraction is the task of splitting the network into several subnetworks with high modularity, namely, more links inside each subnetwork and fewer between subnetworks. Since the role is the aggregation of structural features, nodes may have the same role even if they belong to different communities. More precisely, the roles have no local position or distance limit within a network. In addition, some research has shown that the community structure is fractal, which means that smaller communities' structural features can also be found in larger communities and vice versa (Tsugawa and Ohsaki 2014). Therefore, it is expected that different communities share similar role knowledge (roles' structural features, role distribution, connection patterns between roles, etc.). We propose selecting and labeling the subnetworks with the proper size as the source network and validation network from the community extraction method, shown in line 3 of Algorithm 1. Note that the sizes of the source network and validation network reflect a trade-off between the accuracy of role discovery and the labeling cost and should be determined depending on the particular case. A larger size means more cost for labeling and a higher probability of better accuracy. The target network could be the remaining network or specified subnetworks. By learning the role knowledge in the source network, $f_{mrd} : \theta \rightarrow L$ could be trained to predict the role labels in the validation network and the target network. Here, we choose the optimal $f_{mrd} : \theta \rightarrow L$ by evaluating the accuracy of the predictions for the validation network.

Embedding

Many network embedding methods have been proposed for mapping nodes onto latent feature space while preserving structural identifications, such as the above-mentioned struc2vec, role2vec, and GraphWave. In our previous work, we applied struc2vec to learn the structural representation of nodes, shown in line 2 of Algorithm 1. While struc2vec (Ribeiro et al. 2017) can transform structurally similar nodes into similar vector representations, it does not support weighted or directed networks. Furthermore, as a heuristic method, the computation time and scalability remain issues. Note that our proposed framework is flexible in its ability to switch components with other possible methods. Therefore, in this work, we apply role2vec and GraphWave as network embedding methods in addition to struc2vec. role2vec (Ahmed et al. 2020) maps nodes to roles based on the aggregation of their higher-order structural features. This suggests use of a biased random walk scenario to acquire the roles' context sequence instead of the nodes' sequence. Then, role2vec uses a Skip-Gram model to obtain a distributed representation of roles from the role sequence obtained by the random walk. The distributed representation of a node should be the same as the distributed representation of the mapped role. In other words, all nodes assigned to the same role have the same vector representation. Although role2vec cannot discover multiple roles directly because one node is assigned to one role and the mapped roles may not meet the needs of discovering a wide variety of roles in a real network, it is possible to reduce the computational cost by mapping a large number of nodes to a small number of roles.

GraphWave (Donnat et al. 2018) treats the spectral graph wavelets as probability distributions, gains the empirical characteristic function of the distributions, samples evenly spaced points from the empirical characteristic function, and concatenates the values as the embedding vector of a node. Specifically, since the spectral graph wavelets analogize the Laplacian eigenvalues as temporal frequencies of a signal, GraphWave obtains a diffusion pattern for every node when a Dirac signal is centered around each node. GraphWave involves a scaling parameter s, since it is derived from the spectral graph wavelet. s controls the radius of the neighborhood of each node, so a smaller s constrains the neighborhood near the center node. In contrast, a larger s allows farther nodes to exist within the neighborhood. Namely, a smaller s leads to community-based embedding, while a larger s is suitable for role-based embedding. GraphWave provides a method to automatically find the appropriate range of values for s and a multiple-scaled algorithm to aggregate structural embedding.

The experiments described below were conducted using struc2vec, role2vec, and Graphwave to obtain distributed representations. We show that the proposed method is a flexible framework that allows component replacement.

Domain adversarial learning

Since selecting the training dataset (the source and validation networks) is an arbitrary process, it might cause selection bias. In addition, the structural features, the distribution of roles, and the co-occurrence of roles in the three networks might vary from one to the other, which increases the difficulty of multiple-role discovery. Therefore, following an earlier work (Kikuta et al. 2020; Liu et al. 2021), we use a domain adversarial learning strategy to discover the multiple roles of nodes with high accuracy. The domain adversarial learning strategy (line 12 to 22 in Algorithm 1) consists of two types of neural networks: role-model $f_{mrd}^{\theta^R}$ and discriminator $f_{mrd}^{\theta^D}$, where θ^R and θ^D denote the weights of the role-mode and discriminator, respectively. The purpose of the discriminator is to determine whether a node belongs to the source or the target network. The goal of the role-model is to predict the role labels of a node and to fool the discriminator by making the node embedding representations of the three networks similar to a gradient reversal layer (Ganin et al. 2016). Following the domain adversarial learning strategy, a mini-max game is executed (line 21 in Algorithm 1). The adversarial learning is conducted on a mini-max game on the overall loss:

$$\underset{\theta^R,\theta_{s,v,t}}{min}\ \underset{\theta^D}{max}\big(E_R(V_s;\theta^R,\theta_s)-\lambda J(V;\theta^D,\theta_s,\theta_v,\theta_t)\big). \qquad (1)$$

After converging, the node embedding of the target network is used as input to the role model to predict the corresponding role labels (line 24n Algorithm 1).

Role model for multiple-role discovery The role model is a three-layered neural network whose input is the embedding of the source network and whose output is the probability of the role labels. We apply the multi-label classification scenario to the role model to predict the multiple roles of nodes. Specifically, we use the sigmoid function as the active function of the output layer in the role model. This maps the output of neurons to a range of (0, 1) representing the corresponding label's probability. We select BP-MLL (Zhang and Zhou 2006) as the loss function of the role model.

$$E = \sum_{i=1}^{m} E_i = \sum_{i=1}^{m} \frac{1}{|Y_i||\bar{Y}_i|} \sum_{(k,l)\in Y_i \times \bar{Y}_i} \exp(-(c_k^i - c_l^I)), \tag{2}$$

where m is the number of data and Y_i is the label set of the ith data. $\bar{Y}_i$ is the complement of Y_i, which is the set of labels not coupled with the ith data. c_k^i is the output of the neuron corresponding to the label k of the ith data. BP-MLL earns a penalty when the output of neurons for positive labels is not larger than that of negative labels. Intuitively, not only the output of positive (negative) labels should be larger (smaller) than the threshold, the classifier should be more confident with larger (smaller) output on the positive (negative) labels. In addition to BP-MLL, we also considered binary cross-entropy loss and i-BP-MLL (Grodzicki et al. 2008). Binary cross-entropy loss is a fundamental loss function for the multi-label classification tasks. i-BP-MLL is an extended loss function from BP-MLL. It focuses on determining the value for the threshold and produces a custom threshold for each neuron separately, instead of a global threshold in BP-MLL and binary cross-entropy loss. Consequently, the classifier needs to learn the threshold for each label coupled with the output of each label, which makes the neural network structure complicated and difficult to converge. From our evaluation experiments, the performance of BP-MLL is better than both binary cross-entropy loss and i-BP-MLL. This might be because the binary cross-entropy loss is too simple to train a highly expressive model, while i-BP-MLL is difficult to converge under the domain adversarial learning strategy. Thus, we apply BP-MLL as the loss function of the role model. In Algorithm 1, line 18 to 20 resents the psuedo-code of the role model.

Discriminator variations The discriminator is a three-layered neural network whose input is the embedding of the networks, and whose output is the probability of the domain labels. The role model directly adjusts the embedding representation with a gradient reversal layer according to the discriminator's loss (line 21 in Algorithm 1).

Therefore, the design of the discriminator is exceptionally significant. In earlier work (Liu et al. 2021), we proposed four variations of the discriminator: BPMLL, BPMLL_SD2, BPMLL_SD3, and BPMLL_DD. In this work, we follow up that work (Liu et al. 2021) and present a final empirical conclusion regarding these four variations.

1 **BPMLL**: No discriminator assembled; namely, the domain adversarial learning strategy is not used. The pseudo-code is shown in line 5 to 10 in Algorithm 1.
2 **BPMLL_SD2**: One binary discriminator to determine whether the node is sampled from the source network; in this way, the discriminator does not distinguish between the validation network and the target network. The loss function is

$$J(V;\theta^D,\theta_s,\theta_v,\theta_t) = \frac{1}{|V_s|}\sum_{k\in V_s} \log f_{mrd}^{\theta^D}(\theta_s^k) + \frac{1}{(|V_t|+|V_v|)}\left(\sum_{k\in V_t} \log\left(1 - f_{mrd}^{\theta^D}(\theta_t^k)\right) + \sum_{k\in V_v} \log(1 - f_{mrd}^{\theta^D}(\theta_v^k))\right).$$

3 **BPMLL_SD3**: One ternary discriminator to identify the domain label of the sampled node from the source, target, and validation networks. The loss function is

$$J(V;\theta^D,\theta_s,\theta_v,\theta_t) = \frac{1}{|V_s|}\sum_{k\in V_s}\log f_{mrd}^{\theta^D}(\theta_s^k)_s + \frac{1}{|V_t|}\sum_{k\in V_t}\log f_{mrd}^{\theta^D}(\theta_t^k)_t + \frac{1}{|V_v|}\sum_{k\in V_v}\log f_{mrd}^{\theta^D}(\theta_v^k)_v.$$

$f(\cdot)_s$, $f(\cdot)_t$, and $f(\cdot)_v$ indicate the output of neurons associated with the source, target, and validation networks, respectively.

4 **BPMLL_DD**: Double binary discriminators (one identifies the node's domain label between source network and target network, the other identifies the source network and validation network). The loss function is

$$J(V;\theta^D,\theta_s,\theta_v,\theta_t) = \frac{1}{2}\left(\frac{1}{|V_s|}\sum_{k\in V_s}\log f_{mrd}^{\theta^{D1}}(\theta_s^k) + \frac{1}{|V_t|}\sum_{k\in V_t}\log(1-f_{mrd}^{\theta^{D1}}(\theta_t^k)) + \frac{1}{|V_s|}\sum_{k\in V_s}\log f_{mrd}^{\theta^{D2}}(\theta_s^k) + \frac{1}{|V_v|}\sum_{k\in V_v}\log(1-f_{mrd}^{\theta^{D2}}(\theta_v^k))\right).$$

The pseudo-code of variations of the discriminator is given in line 12 to 22 in Algorithm 1, and $J(\cdot)$ in line 16 indicates the above loss function of each variation. Different variations represent different scenarios to reduce the selection bias discussed above. Specifically, BPMLL_SD2 treats the target and validation networks as a single integrated network, tries to reduce the distribution difference between the integrated network and the source network, and is expected to be effective for cases where the target network and validation network are highly similar; BPMLL_SD3 is derived naturally from the idea of taking the validation network into account; BPMLL_DD strengthens the leading position during the embedding adjustment by aligning the target/validation networks to the source network separately.

Overall, the input and latent layers of role model and discriminator apply the relu active function and dropout strategy, and the optimization function is adam with a weight decay of 0.0001. The hyperparameters of adversarial learning are grid searched: training epochs from (500, 1000, 5000), dropout rate from (0.25, 0.5), and learning rates from (0.01, 0.001, 0.0001, and 0.00001).

Computational complexity

The computational complexity of the proposed framework can be divided into three stages: the complexity of network embedding O_{emb}, the complexity of the role model O_{role}, and the complexity of the discriminator O_{disc}. According to one study (Ribeiro et al. 2017), the complexity of struc2vec is $O_{struc2vec} = O(k|V|^3)$, where k is the diameter of the input network and $|V|$ represents the number of nodes. With these optimizations, the complexity can be reduced further to $O_{struc2vec} = O(|V|^3)$. In another work (Donnat et al. 2018), the complexity of GraphWave is $O_{GraphWave} = O(K \times |E|) = O(|E|)$, where K denotes the Chebyshev polynomial approximation's order and $|E|$ represents the number of edges. Based on reported calculations (Ahmed et al. 2020), the

Table 1 Statistics of Wikipedia network and communities

Name	# Nodes	# Edges	# Roles
Wiki_all	4777	184,812	40
Wiki_c1	958	7946	18
Wiki_c2	1633	8550	18
Wiki_c3	442	1982	18

complexity of role2vec is roughly $O_{role2vec} = O(|V|^3)$. The complexity of the role model is $O_{role} = O(|V_s| \times t \times s \times (i \times j + j \times k + k \times |L|))$, where $|V_s|$ is the number of nodes in the source network, t is the number of training epochs, s is the number of hyper-parameter's combinations, i, j, k are the numbers of neurons in the corresponding layers, and $|L|$ is the number of role labels. Normally i, j, k are set to $8 \times |L|$, $4 \times |L|$, and $2 \times |L|$, respectively. Thus, $O_{role} = O(42 \times |V_s| \times t \times s \times |L|^2))$. Since t and s are constants and $|L| \ll |V|$, then $O_{role} = O(|V|^3)$. Similarly, the complexity of the discriminator is $O_{disc} = O((|V_s| + |V_t| + |V_v|) \times t \times s \times (i \times j + j \times k + k \times 3))$, where $|V_t|$ and $|V_v|$ are the numbers of nodes in the target network and the validation network and i, j, k are the same as those in the role model. $O_{disc} = O(40 \times (|V_s| + |V_t| + |V_v|) \times t \times s \times |L|^2)) = O(|V|^3)$. The total complexity of our proposed framework $O_{all} = O_{emb} + O_{role} + O_{disc} = O(|V|^3)$ theoretically. The total complexity could be further reduced by applying a Bayesian optimization method and successive halving search.

Experiment

Datasets of real networks

In order to make a fair comparison to our former work (Liu et al. 2021), we use the Wikipedia[1] and Blogcatalog datasets (Zhang et al. 2019) as experimental data. The Wikipedia network is a word co-occurrence network with 4777 nodes, 184,812 edges, and 40 different node role labels (part-of-speech labels for words). We treat the part of speech as the role of words under the supposition that the part of speech correlates to the network structure. Words with different parts of speech have different connection patterns. For example, nouns are more often co-occurrent with verbs, while transitive verbs are more co-occurrent with nouns, pronouns, and adverbs. The words 'water' and 'fire' have both noun and transitive verb labels. The connection patterns of the two words contain characteristics of both nouns and transitive verbs. Thus, the local structure of 'water' and 'fire' might be similar, even though the two words are located far away in the Wikipedia network. For the entire network (Wiki_all), community extraction was performed, and three sub-networks (Wiki_c1, Wiki_c2, and Wiki_c3) were randomly selected from the results as the source network, target network, and validation network, respectively. The statistics for each network are shown in Table 1, and the experimental data for the six datasets are shown in Table 2. In addition, 18 role labels common to the three

[1] http://www.mattmahoney.net/dc/text.html.

Table 2 Wikipedia datasets

No.	Source	Target	Validation
Wiki-1	Wiki_c1	Wiki_c2	Wiki_c3
Wiki-2	Wiki_c1	Wiki_c3	Wiki_c2
Wiki-3	Wiki_c2	Wiki_c1	Wiki_c3
Wiki-4	Wiki_c2	Wiki_c3	Wiki_c1
Wiki-5	Wiki_c3	Wiki_c1	Wiki_c2
Wiki-6	Wiki_c3	Wiki_c2	Wiki_c1

Table 3 Statistics of Blogcatalog network and communities

Name	# Nodes	# Edges	# Roles
Blog_all	10,312	333,983	39
Blog_c1	3271	69,105	38
Blog_c2	2941	42,480	38
Blog_c3	3302	66,807	38

Table 4 Blogcatalog datasets

No.	Source	Target	Validation
Blog-1	Blog_c1	Blog_c2	Blog_c3
Blog-2	Blog_c1	Blog_c3	Blog_c2
Blog-3	Blog_c2	Blog_c1	Blog_c3
Blog-4	Blog_c2	Blog_c3	Blog_c1
Blog-5	Blog_c3	Blog_c1	Blog_c2
Blog-6	Blog_c3	Blog_c2	Blog_c1

subnetworks, such as abstract noun, concrete noun, pronoun, and adjective, were used for training and prediction.

The Blogcatalog network is an interaction network among bloggers in an online blogging service with 10,312 nodes (bloggers), 333,983 edges (interactions between bloggers), and 39 role labels indicating the hobbies of the bloggers. Since that blogger may have multiple hobbies, the Blogcatalog dataset is multilabel. We assume that the hobbies correlate with the network structure, such as the bloggers who share the same hobby are not necessarily connected; this is proven by simply calculating the average number of connected components in the sub-network where the nodes share the specified label. For the Blogcatalog dataset, it is 30.8, which means that the nodes that share the specified label are separated into 30.8 components. Similar to the Wikipedia dataset, for the whole network (Blog_all), we extracted and randomly selected three sub-networks, Blog_c1, Blog_c2, and Blog_c3 (Table 3), and assigned them as the source, target, and validation network to form six datasets (Table 4.)

Baselines

We evaluated the effectiveness of our proposed framework by the accuracy of multiple-role discovery using real-world networks. In a previous work (Liu et al. 2021), we already tested the feasibility of the domain adversarial learning method for multiple-role discovery specified with struc2vec. The results show that the proposed method (Liu et al. 2021) could achieve higher accuracy in most cases.

In this paper, we mainly focus on evaluating the creation of the training dataset constructed by community extraction. We applied random sampling to create the training dataset (the equivalent of the source and validation network) as the comparison method. Specially, we first learned the embedding of all nodes in a real network, and then we sampled the same number of nodes randomly as the source (validation) network to form the source (validation) dataset. We used the same nodes in the target network as the target data to control the variables. Note that we split the whole real network into subnetworks and selected two appropriate subnetworks as the source and validation networks; the target network could be the remaining network $G(V\backslash\{V_s, V_v\}, E)$ or specified subnetworks. If the target network were the remaining network $G(V\backslash\{V_s, V_v\}, E)$, random sampling would be impossible, since it degrades to a problem of dividing a set into two groups. Therefore, we selected one subnetwork as the target network in this experiment. Note that this is just for the evaluation experiment: In practical cases, the target network could be the remaining network.

After creating the training dataset, we feedforwarded it to three conventional multi-label classification scenarios to predict the multiple roles: Binary Relevance (BR) (Zhang and Zhou 2013; Zhang et al. 2018), Label Powerset (LP) (Zhang and Zhou 2013; Herrera et al. 2016), and MLkNN (Zhang and Zhou 2007). Specifically, in the BR scenario, several binary SVM classifiers (Vapnik 1997) for each label were trained, and grid search was applied for hyperparameter tuning between radial basis function kernel and linear kernel. In the LP scenario, a Random Forest multi-class classifier (Breiman 2001) corresponding to the label's combination pattern was trained, and grid search was applied for hyperparameter tuning in the same way as with the BR scenario. The hyperparameters for the Random Forest multi-class classifier were searched as follows: the number of trees from (20, 30, 50) and the quality measuring function from Gini and entropy. Since MLkNN is a multi-label classification method, accuracy verification was performed directly on each dataset. In the datasets constructed by random sampling, only the roles present in the training and validation data could be predicted, so the range of roles could not be fully controlled. The size of the role set varied from 16 to 20 in different samplings.

Justification confirmation of community extraction

We propose using community extraction methods to separate a real network into subnetworks for the source and validation networks. To confirm its justification, we investigated the difference between community extraction and random sampling from the viewpoints of node coverage and label powerset coverage. Node coverage is a criterion to evaluate the labeling cost for the source and validation networks. Since, in most cases, it is necessary to check a node's neighbors to label its roles, the node coverage is the number of nodes that must be checked when labeling the roles of the nodes in a subnetwork. For example, coverage of k-hop nodes is possible, namely, the nodes in a subnetwork and their k-hop neighbors. Here, we investigate 1-hop node coverage between the subnetwork selected through community extraction and random sampling. For community

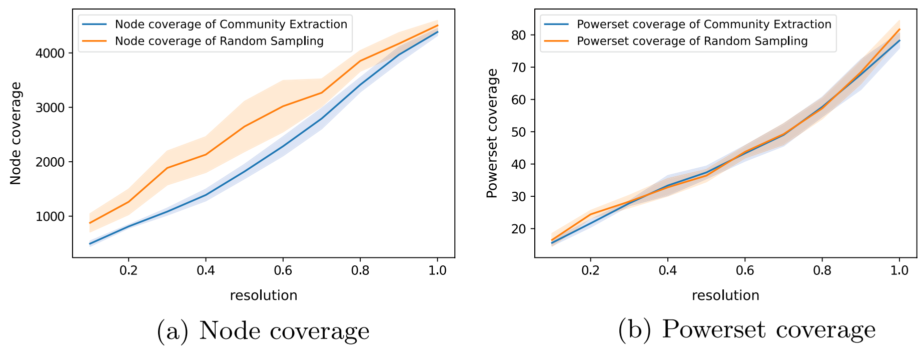


Fig. 2 Node coverage and powerset coverage between community extraction and random sampling

extraction, we applied the Louvain method to the Wikipedia dataset (Wiki_all) using different resolutions ranging from 0.1 to 1, and we selected the four largest subnetworks to calculate the average 1-hop node coverage. Resolution represents time as described previously (Lambiotte et al. 2008), leveraging the community's size: the higher resolution for a larger size. Then for random sampling, we randomly sampled the same number of nodes as the four subnetworks selected from community extraction and calculated the average of their 1-hop node coverage. For each resolution, we conducted ten experiments and plotted the average and standard deviation in Fig. 2a.

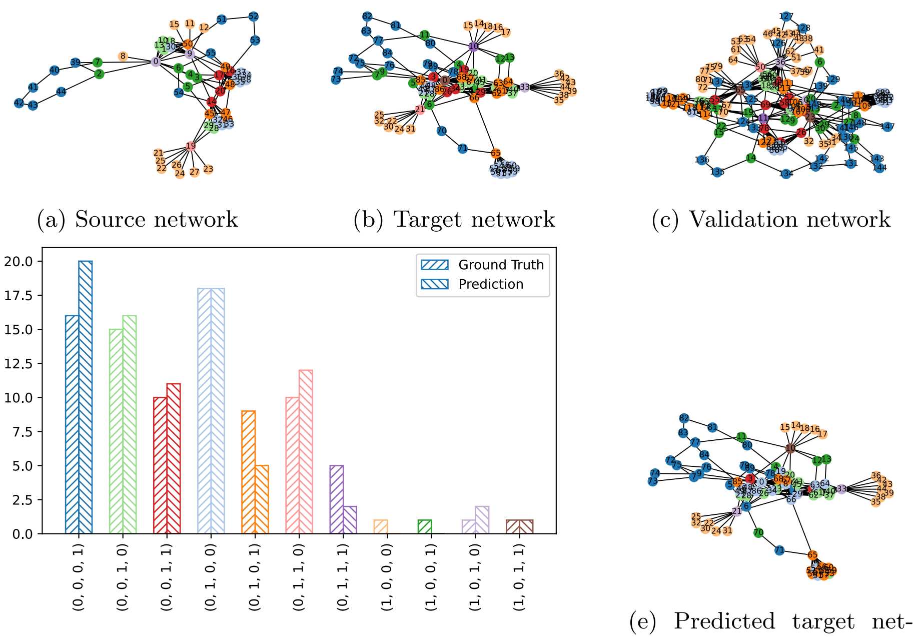
The horizontal axis is the resolution of the community extraction method, and the vertical axis is the 1-hop node coverage. At all of the resolutions, the node coverage of community extraction is lower than that of random sampling, indicating community extraction's validity in reducing the labeling cost. A possible reason for the decrease in the difference between community extraction and random sampling as the resolution increases is the dense structure of the original real network.

Moreover, it is worth confirming the labels' diversity in the selected subnetworks because this is related to the quality of the training data. Therefore, to evaluate the labels' diversity, we use the powerset coverage of labels, which is the number of label combinations for all nodes in the selected subnetworks. Like node coverage, we compare the average powerset coverage of the four largest communities at different resolutions and those of the randomly sampled nodes of the same sizes. The result is shown in Fig. 2a, which demonstrates that both community extraction and random sampling have the same level of diversity in the labels' combinations. Overall, creating the training datasets by using community extraction methods can reduce the labeling cost without harming the quality of label diversity.

Case study on toy dataset

We conducted a case study on a toy dataset in Fig. 3 to confirm whether the proposed framework can discover multiple roles for a single node. We used three types of network topologies (mesh, ring, and star) and connected them with each other by nodes randomly selected to generate networks as shown in Fig. 3a–c.

The roles for each node are from their original network topologies, i.e., star-center, star-periphery, mesh, and ring; nodes contained in different topologies have multiple roles. Nodes are colored based on their combinations of roles. We created three different-sized



(a) Source network (b) Target network (c) Validation network

(d) Histograms of label combinations (e) Predicted target network

Fig. 3 Toy dataset and results

networks and assigned them as source, target, and validation networks. We used the struc2vec embeddings and BPMLL_SD2 to predict the role labels in the target network, and the result is shown in Fig. 3d, e.

The histograms of the ground truth for the role combinations and that of prediction are shown in Fig. 3d. The horizontal axis shows the combinations of the four roles, colored the same as the nodes' colors in Fig. 3b, and the label is a set of four binary numbers associated with star-center, star-periphery, mesh, and ring. For instance, the label of the bars on the leftmost side (0, 0, 0, 1) represents the nodes with the role of ring; (1, 0, 1, 1) on the rightmost side indicates the multiple roles of star-center, mesh, and ring. The vertical axis is the number of nodes. The ground truth value is plotted on the left side for each combination, and the prediction is on the right side. The prediction results show that the proposed framework was able to approximate the tendency of the ground truth and could predict the multiple roles for single nodes, even for under-represented combinations such as star-periphery, mesh, ring, and purple bars denoted by the (0, 1, 1, 1) combination. Moreover, by comparing the nodes' colors in Fig. 3b (truth labels) and Fig. 3e (prediction labels), we find that 80% of the nodes with single roles (such as star-periphery for node 14, mesh for node 46, and ring for node 69) were predicted correctly; 55% of nodes with double roles (such as star-periphery and ring for node 4, star-periphery and mesh for node 20, star-center and mesh for node 33, mesh and ring for node 65) were predicted exactly; node 3 with three roles (star-periphery, mesh, and ring) was classified appropriately.

Experiments on real networks

Comparison between the previous (STV) and the proposed methods

We conducted experiments to evaluate the performance of the previous method (STV) on the Wikipedia dataset (Wiki-1 in Table 2). As mentioned above, the STV method was

Table 5 Results on Wiki-1 with GraphWave embeddings

Method	Accuracy	Precision	Recall	F-measure
STV method	0.180	0.206	0.460	0.276
Proposed method	0.380	0.444	0.586	0.503

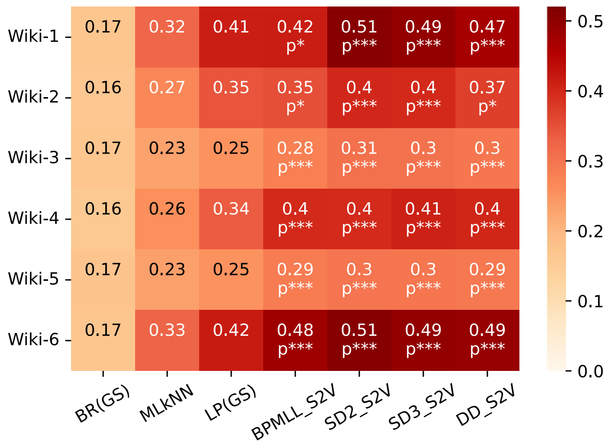
proposed for single role discovery. Thus, we applied the binary relevance strategy to the STV method by (1) training an STV model for each role, (2) assembling the results of the best performances to form an $18 \times |V_t|$ matrix, and (3) evaluating the matrix using F-measure through comparison with that of the proposed framework. The results are shown in Table 5. Note that the purpose of this experiment is to confirm the validity of the proposed method apart from the STV method; considering the calculation cost, we only show the average F-measure of two trials on GraphWave embeddings.

The result show that the proposed framework's performance was superior to that of the STV method with binary relevance in all criteria, indicating the proposed framework's necessity. Moreover, we observed that the STV method tends to predict the under-represented labels as non-existent in all nodes, which may be caused by neglecting the relationships between roles.

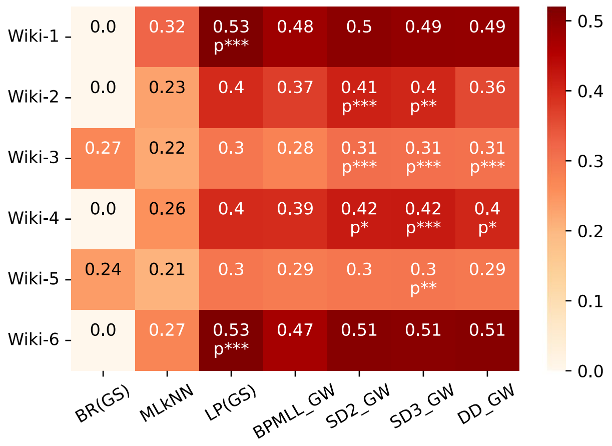
Comparision between proposed framework and baselines on different embeddings

We conducted experiments on both the proposed framework and baselines of different embeddings of Wiki-1 to Wiki-6 and Blog-1 to Blog-6. For each method, ten experiments were conducted, and the average F-measures are shown in Fig. 4 for the Wikipedia datasets and Fig. 5 for the Blogcatalog datasets on embeddings learned by struc2vec, GraphWave, and role2vec. In each figure, the three columns from left to right are the comparison methods (random sampling): BR, MLkNN, and LP; the four columns on the right side are the variations of the proposed method in "Domain adversarial learning" @@section: BPMLL (BPMLL_S2V, BPMLL_GW, and BPMLL_R2V in figures), BPMLL_SD2 (SD2_S2V, SD2_GW, and SD2_R2V in figures), BPMLL_SD3 (SD3_S2V, SD3_GW, and SD3_R2V in figures), and BPMLL_DD (DD_S2V, DD_GW, and DD_R2V in figures). The average F-measures of the method on each embedding are shown in the related cell. The p*, p**, and p*** below the numbers represent the p-values' ranges ($p > 0.05$, $0.05 \geq p > 0.01$, and $p \leq 0.01$, respectively) of the independent samples' t-tests between the proposed and the comparison methods. For the proposed methods BPMLL, BPMLL_SD2, BPMLL_SD3, and BPMLL_DD, the t-test is conducted with the comparison method having the highest average F-measure; for the comparison methods BR(GS), MLkNN, and LP(GS), vice versa. The p-value range is shown only if the t-statistic is over zero.

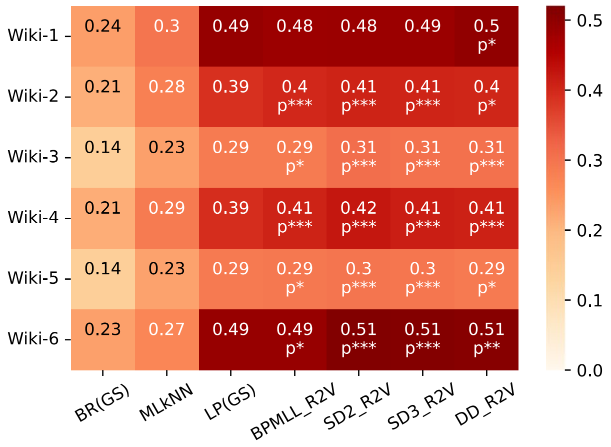
The results of the F-measure from the experiment using the Wikipedia dataset embeddings obtained with struc2vec are shown in Fig. 4a. All four proposed method variations achieved higher accuracy than the comparison methods; 21 out of 24 cases had a p-value ≤ 0.01 in the t-test and showed significant differences from all of the comparison methods. Among them, variations with discriminators achieved higher accuracy than BPMLL (without any discriminator), which shows that domain adversarial learning alleviates the bias brought by community selection. BPMLL_SD2 and BPMLL_SD3 achieved the



(a) Results of struc2vec embeddings of Wikipedia

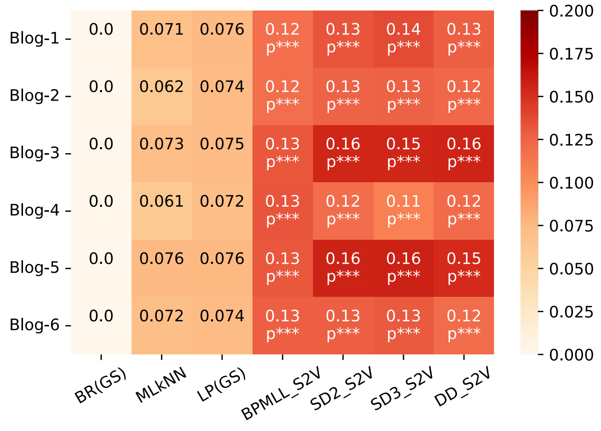


(b) Results of GraphWave embeddings of Wikipedia

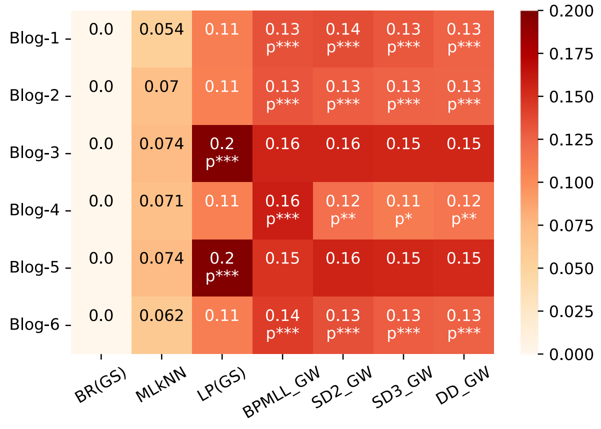


(c) Results of role2vec embeddings of Wikipedia

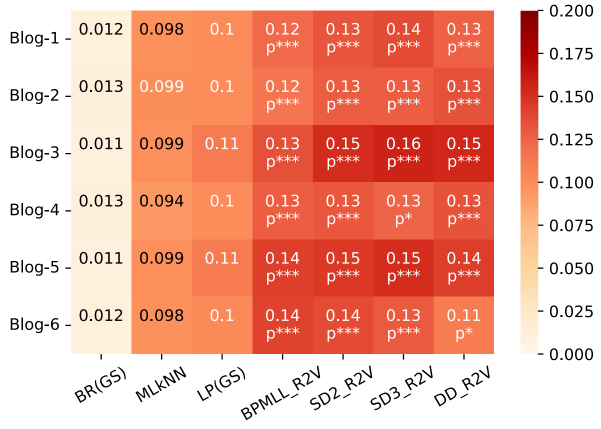
Fig. 4 Results of Wikipedia datasets



(a) Results of struc2vec embeddings of Blogcatalog



(b) Results of GraphWave embeddings of Blogcatalog



(c) Results of role2vec embeddings of Blogcatalog

Fig. 5 Results of Blogcatalog datasets

highest accuracy; the mean, variance, and the number of outliers varied with each dataset; however, no significant trends were found overall. The highest performance from all of the methods ranges from 0.3 to 0.51 (precisely, 0.51 for Wiki-1 and Wiki-6, 0.4 for Wiki-2 and Wiki-4, and 0.31 for Wiki-3 and Wiki-5). From Table 2, it could be inferred that the best performance of one dataset has a very strong co-relationship with the target network; for instance, Wiki-1 and Wiki-6 share the same target network Wiki_c2. However, for the same target network, the exchange of the source and validation network doesn't change the accuracy, indicating that the accuracy rarely has a relationship with the difference between the source and validation network.

Figure 4b shows the experiment's results using the embeddings obtained with GraphWave. The proposed methods with discriminator achieved the highest accuracy in Wiki-2, Wiki-3, Wiki-4, and Wiki-5 with a significant difference, while LP(GS) achieved the highest accuracy in Wiki-1 and Wiki-6. The possible reason for LP(GS)'s performance on Wiki-1 and Wiki-6 might be that GraphWave is not associated with a random walk and can create embeddings without fluctuations, which leads to an easier prediction task. BPMLL's results were lower than those of the other proposed methods with discriminators in all cases, indicating that the adversarial learning framework helped to increase the accuracy of this embedding. Overall, BPMLL_SD3 performed the best among all methods. The pattern of the same target network getting the same level of accuracy could also be found in these results.

Figure 4c shows the results on role2vec embeddings. Like the results of struc2vec in Fig. 4a, the four proposed method variations achieved higher accuracy than the comparison methods in 21 out of 24 cases; 14 cases had a p-value of ≤ 0.01 in the t-test. The proposed methods achieved the highest accuracy in all datasets. BPMLL_SD2 and BPMLL_SD3 achieved the highest accuracy except on Wiki-1; BPMLL_DD had the highest but not significantly different F-measure to LP(GS) on Wiki-1, indicating that the result of BPMLL_DD fluctuating among the ten experiments, showing that the two discriminators in BPMLL_DD increased the potential of the method's predictive ability and added coverage difficulties in training. Moreover, the tendency of the highest accuracy on different datasets is similar to the results on struc2vec and GraphWave.

The results of the F-measure from the experiment using the Blogcatalog dataset embeddings are shown in Fig. 5. This figure should be read in the same way as Fig. 4. The overall result pattern of the Blogcatalog dataset is similar to that of the Wikipedia dataset. However, the F-measures are generally low, up to 0.16, indicating the difficulty of prediction, which might be caused by a large number of labels (38 labels against 18 labels in the Wikipedia dataset) and the sparseness of labels (density of 0.036 against 0.080 in the Wikipedia dataset).

The results of experiments on struc2vec embeddings in Fig. 5a show that, unlike the Wikipedia datasets, BPMLL achieved the highest accuracy in Blog-4 and Blog-6. This might be explained by how the increased prediction complexity for more labels in Blogcatalog datasets also made it more challenging for the discriminators and role models to converge. Moreover, the tendency of the highest accuracy on different datasets being related to their target networks could be observed for Blog-3 and Blog-5, which share the same target network (i.e., Blog_c1). Figure 5b shows the F-measure results on GraphWave embeddings. The proposed methods reached higher accuracy than the comparison methods, with a significant difference in 15 out of 24 cases, while LP(GS) achieved the highest accuracy in Blog-3 and Blog-5. Reflecting the results of GraphWave embeddings on the Wikipedia datasets in Fig. 4b, LP(GS) achieved the highest accuracy in Wiki-1 and Wiki-6, which share the same target network of Wiki_c2, similar to the results on the Blogcatalog datasets. This phenomenon supports the assumption that GraphWave creates embeddings without fluctuations, which leads to an easier prediction task. Thus, more straightforward methods such as LP(GS) and BPMLL can achieve higher accuracy. Figure 5c shows the experiment results on role2vec embeddings. Again, the results support the hypothesis that the highest accuracy on different datasets tend to be related to the target network. Overall, the results of role2vec are similar to those of struc2vec in the performances of the proposed methods, patterns of different target network combinations, and highest accuracy on each dataset, demonstrating that embeddings obtained by role2vec and struc2vec have a similar affect on the downstream task, which might be caused by the use of random walk.

Discussion

In the three experiments with various embeddings, the performance of the proposed methods ranged from 0.3 to 0.55 on the Wikipedia datasets and 0.12 to 0.18 on the Blogcatalog datasets. The numbers of labels are around 18 for the Wikipedia datasets and 38 for the Blogcatalog datasets, as mentioned in Tables 1 and 3. This difference between the two datasets indicates the difficulty of multi-label classification, thus leading to the difference in accuracy. Furthermore, the F-measure results of the two datasets are relatively low, which may be restricted by the co-relationship between the network structure features (words' co-occurrence network structure for Wikipedia and blogger's interaction network structure for Blogcatalog) and the roles' information (part-of-speech labels for words in Wikipedia and blogger's hobbies in Blogcatalog), which indicates that predicting functional role labels only from structure features has limitations. We believe multiple functional role discovery would achieve better performance coupled with the additional attribute information of nodes or edges. In addition, for each dataset (Wiki-1 to Wiki-6 and Blog-1 to Blog-6), the best performances with three embeddings were at the same level but with slight fluctuations, indicating the role distribution bias and structural bias for different subnetworks. Among the Wikipedia datasets, Wiki-3 and Wiki-5 achieved the lowest F-measure values; according to Table 2, both combinations have the same target network Wiki_c1. In contrast, in the Blogcatalog datasets, the highest accuracy was achieved by Blog-3 and Blog-5, which share the same target network (Blog_c1). We analyzed the low (high) performance as the gap between Wiki_c1 (Blog_c1) and the other two subnetworks. The result indicates that the selection of the subnetworks (especially the target network) does affect the prediction performance significantly, even though the proposed framework outperforms random sampling. Optimizing the subnetworks' selection remains a future task.

For two real networks, in the experiments with struc2vec embeddings and role2vec embeddings, the proposed variations with a discriminator (BPMLL_SD2, BPMLL_SD3, or BPMLL_DD) achieved the highest accuracy. In the experiments with GraphWave embeddings, the proposed variations (especially BPMLL_SD2 and BPMLL) outperformed others in half of the cases. Thus, overall, the proposed method (BPMLL_SD2, BPMLL_SD3, and BPMLL_DD) showed its superiority over various embedding methods. These results indicate that the proposed framework is flexible for discovering multiple roles in real networks in different embedding scenarios. For different embedding methods and datasets, BPMLL_SD2, BPMLL_SD3, and BPMLL_DD show different tendencies of accuracy in mean, variance, and number of outliers. Generally, BPMLL_SD2 performed best in more cases, which shows its stability. In contrast, BPMLL_DD produced unstable results with a wide range of accuracy in repetitions, which might be caused by the high expressive ability and considerable convergence difficulty of using two discriminators. Note that in the datasets constructed by random sampling, only the roles present in the training and validation data could be predicted, so the range of roles could not be fully controlled, which is a critical limitation of the random sampling approach. To sum up, the results indicate that one can create training datasets by the community extraction method, choose the appropriate embedding method, and select the BPMLL_SD2 or BPMLL_SD3 variation in a practical role discovery task.

In this work, although selecting the embedding method is not our research purpose, we did find some tendencies of multiple-role discovery results with different embedding methods. Compared with struc2vec, role2vec's role discovery results show minor variance in almost all cases. This might be due to multiple nodes sharing the same embedding if they belong to the same role category. Thus, the input for classifiers is consistent and less biased, leading to a stable prediction result. GraphWave further shrank the range of variance, especially for the LP (random sampling) method. A possible reason might be that GraphWave is the only embedding method without any random walk process and thus could produce solid embeddings.

We found that the highest accuracy for each dataset with different embedding methods reached the same level within two real networks. This phenomenon implies that the highest accuracy indicates the maximum mutual information between the structural features and the multiple roles. Note that the role is defined by the aggregation of structural features or functions. Moreover, for practical use in real networks, one might be more interested in the nodes' functions rather than the structural features. This work shows that mining the structural features is a promising way to discover the functional role labels.

Overall, this work proposed a flexible framework for multiple role discovery of real networks. Using community extraction to create the training dataset is proven to be valid and feasible for real networks. Compared with the STV method and other multiclass classification strategies, the adversarial learning scenario and the loss function of the role model make the proposed framework robust against the biases of role distribution and network structure.

Conclusion

This study proposed a flexible framework for discovering multiple roles in real networks. We conducted intensive experiments on two real networks, one a lexical co-occurrence network and the other a social network, to check the generalization capability of the proposed framework. We tested three embedding methods and showed the proposed framework's feasibility for various embedding components. We also proposed an approach for constructing training data by community extraction for the role discovery task in real networks, evaluated the approach against the straightforward approach (random sampling), and demonstrated its superiority in most cases. Through experiments on the Wikipedia network and the Blogcatalog network, we demonstrated that it is promising to predict functional role labels from structural features. We also confirmed that domain adversarial learning could provide stable learning and prediction when constructing source networks with community extraction, even when selection bias exists, thus further demonstrating the potential of the proposed framework for real-world applications.

Future tasks for this study include optimizing the domain adversarial learning framework and achieving more efficient hyperparameter tuning. Specifically, the computational complexity of our proposed framework is relatively high, which limits its scalability. Improving the efficiency of the hyperparameter tuning (such as a Bayesian optimization method or successive halving search) could be expected to reduce the computational complexity. Furthermore, we are considering updating domain adversarial learning with state-of-the-art strategies to enhance prediction performance and scalability. Furthermore, it would be worthwhile to investigate the influence of the community extraction algorithms and to optimize the subnetworks' selection for stable prediction results.

Acknowledgements
This work was supported by JST SPRING, Grant Number JPMJSP2108.

Author contributions
SL organized research, built the framework, conducted experiments, and analyzed the results. FT and MN designed the work and analyzed the results. SU partially designed the work. All authors read and approved the final manuscript.

Author information
Shu Liu is currently in the doctoral program at the School of Engineering, the University of Tokyo. His work focuses specifically on mining nodes' and edges' structural information for complex networks and the network embedding methods.

Funding
This work was supported by JST SPRING, Grant Number JPMJSP2108.

Availability of data and materials
The implementation codes and data are available at Github (https://github.com/liushu2019/MRD4Real).

Declarations

Ethics approval and consent to participate
Not applicable

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 25 March 2022 Accepted: 19 September 2022
Published online: 28 September 2022

References

Ahmed N, Rossi RA, Lee J, Willke T, Zhou R, Kong X, Eldardiry H (2020) Role-based graph embeddings. IEEE Trans Knowl Data Eng 34(5):2401–2415
Airoldi EM, Blei DM, Fienberg SE, Xing EP (2008) Mixed membership stochastic blockmodels. J Mach Learn Res 9:1981–2014
Akoglu L, Tong H, Koutra D (2015) Graph based anomaly detection and description: a survey. Data Min Knowl Discov 29(3):626–688
Arabie P, Boorman SA, Levitt PR (1978) Constructing blockmodels: how and why. J Math Psychol 17(1):21–63
Baker M, Croft W (2017) Lexical categories: legacy, lacuna, and opportunity for functionalists and formalists. Annu Rev Linguist 3:179–197
Batagelj V, Mrvar A, Ferligoj A, Doreian P (2004) Generalized blockmodeling with Pajek. Metodoloski zvezki 1(2):455–467
Bhagat S, Cormode G, Muthukrishnan S (2011) Node classification in social networks. In: Aggarwal C (ed) Social network data analytics. Springer, pp 115–148
Blondel VD, Guillaume J-L, Lambiotte R, Lefebvre E (2008) Fast unfolding of communities in large networks. J Stat Mech Theory Exp 2008(10):10008
Brandes U, Lerner J (2010) Structural similarity: spectral methods for relaxed blockmodeling. J Classif 27(3):279–306
Breiman L (2001) Random forests. Mach Learn 45(1):5–32
Burt RS (1976) Positions in networks. Soc Forces 55(1):93–122

Cai H, Zheng VW, Chang KC-C (2018) A comprehensive survey of graph embedding: problems, techniques, and applications. IEEE Trans Knowl Data Eng 30(9):1616–1637
Donnat C, Zitnik M, Hallac D, Leskovec J (2018) Learning structural node embeddings via diffusion wavelets. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, pp 1320–1329
Ganin Y, Ustinova E, Ajakan H, Germain P, Larochelle H, Laviolette F, Marchand M, Lempitsky V (2016) Domain-adversarial training of neural networks. The journal of machine learning research 17(1):2096–2030
Goyal P, Ferrara E (2018) Graph embedding techniques, applications, and performance: a survey. Knowl-Based Syst 151:78–94
Grodzicki R, Mańdziuk J, Wang L (2008) Improved multilabel classification with neural networks. In: International conference on parallel problem solving from nature. Springer, pp 409–416
Grover A, Leskovec J (2016) node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pp 855–864
Herrera F, Charte F, Rivera AJ, del Jesus MJ (2016) Multilabel classification. Springer, Cham, pp 17–31. https://doi.org/10.1007/978-3-319-41111-8_2
Holland PW, Laskey KB, Leinhardt S (1983) Stochastic blockmodels: first steps. Social networks 5(2):109–137
Jin J, Heimann M, Jin D, Koutra D (2022) Towards understanding and evaluating structural node embeddings. ACM Trans Knowl Discov Data 16:58–15832
Kikuta S, Toriumi F, Nishiguchi M, Liu S, Fukuma T, Nishida T, Usui S (2020) Framework for role discovery using transfer learning. Appl Netw Sci 5(1):1–19
Kumar A, Singh SS, Singh K, Biswas B (2020) Link prediction techniques, applications, and performance: a survey. Physica A 553:124289
Lambiotte R, Delvenne JC, Barahona M (2008) Laplacian dynamics and multiscale modular structure in networks. arXiv preprint arXiv:0812.1770
Liu S, Toriumi F, Nishiguchi M, Usui S (2021) Multiple role discovery in complex networks. In: International conference on complex networks and their applications. Springer, pp 415–427
Lü L, Zhou T (2011) Link prediction in complex networks: a survey. Physica A 390(6):1150–1170
Mahadevan P, Krioukov D, Fomenkov M, Dimitropoulos X, Claffy K, Vahdat A (2006) The internet as-level topology: three data sources and one definitive metric. ACM SIGCOMM Comput Commun Rev 36(1):17–26
Nowicki K, Snijders TAB (2001) Estimation and prediction for stochastic blockstructures. J Am Stat Assoc 96:1077–1087. https://doi.org/10.1198/016214501753208735
Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining, pp 701–710
Ribeiro LF, Saverese PH, Figueiredo DR (2017) struc2vec: learning node representations from structural identity. In: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, pp 385–394
Rossi RA, Gallagher B, Neville J, Henderson K (2013) Modeling dynamic behavior in large evolving graphs. In: Proceedings of the sixth ACM international conference on web search and data mining, pp 667–676
Rossi RA, Ahmed NK (2014) Role discovery in networks. IEEE Trans Knowl Data Eng 27(4):1112–1131
Rossi RA, Jin D, Kim S, Ahmed NK, Koutra D, Lee JB (2020) On proximity and structural role-based embeddings in networks: misconceptions, techniques, and applications. ACM Trans Knowl Discov Data 14(5):1–37
Rossi R, Fahmy S, Talukder N (2013) A multi-level approach for evaluating internet topology generators. In: 2013 IFIP networking conference. IEEE, pp 1–9
Tang J, Aggarwal C, Liu H (2016) Node classification in signed social networks. In: Proceedings of the 2016 SIAM international conference on data mining. SIAM, pp 54–62
Tang J, Qu M, Wang M, Zhang M, Yan J, Mei Q (2015) Line: Large-scale information network embedding. In: Proceedings of the 24th international conference on world wide web, pp 1067–1077
Traag VA, Waltman L, Van Eck NJ (2019) From Louvain to Leiden: guaranteeing well-connected communities. Sci Rep 9(1):1–12
Tsugawa S, Ohsaki H (2014) Emergence of fractals in social networks: analysis of community structure and interaction locality. In: 2014 IEEE 38th annual computer software and applications conference, pp 568–575. https://doi.org/10.1109/COMPSAC.2014.80
Vapnik VN (1997) The support vector method. In: International conference on artificial neural networks, pp 261–271. Springer
Zhang M-L, Zhou Z-H (2006) Multilabel neural networks with applications to functional genomics and text categorization. IEEE Trans Knowl Data Eng 18(10):1338–1351
Zhang M-L, Zhou Z-H (2007) ML-KNN: a lazy learning approach to multi-label learning. Pattern Recognit 40(7):2038–2048
Zhang M-L, Zhou Z-H (2013) A review on multi-label learning algorithms. IEEE Trans Knowl Data Eng 26(8):1819–1837
Zhang M-L, Li Y-K, Liu X-Y, Geng X (2018) Binary relevance for multi-label learning: an overview. Front Comput Sci 12(2):191–202
Zhang J, Dong Y, Wang Y, Tang J, Ding M (2019) Prone: fast and scalable network representation learning. In: IJCAI, vol 19, pp 4278–4284

Publisher's Note