A Survey on Role-Oriented Network Embedding

Pengfei Jiao, Xuan Guo, Ting Pan, Wang Zhang, and Yulong Pei

Abstract—Recently, Network Embedding (NE) has become one of the most attractive research topics in machine learning and data mining. NE approaches have achieved promising performance in various of graph mining tasks including link prediction and node clustering and classification. A wide variety of NE methods focus on the proximity of networks. They learn community-oriented embedding for each node, where the corresponding representations are similar if two nodes are closer to each other in the network. Meanwhile, there is another type of structural similarity, i.e., role-based similarity, which is usually complementary and completely different from the proximity. In order to preserve the role-based structural similarity, the problem of role-oriented NE is raised. However, compared to community-oriented NE problem, there are only a few role-oriented embedding approaches proposed recently. Although less explored, considering the importance of roles in analyzing networks and many applications that role-oriented NE can shed light on, it is necessary and timely to provide a comprehensive overview of existing role-oriented NE methods. In this review, we first clarify the differences between community-oriented and role-oriented network embedding. Afterwards, we propose a general framework for understanding role-oriented NE and a two-level categorization to better classify existing methods. Then, we select some representative methods according to the proposed categorization and briefly introduce them by discussing their motivation, development and differences. Moreover, we conduct comprehensive experiments to empirically evaluate these methods on a variety of role-related tasks including node classification and clustering (role discovery), top-k similarity search and visualization using some widely used synthetic and real-world datasets. Finally, we further discuss the research trend of role-oriented NE from the perspective of applications and point out some potential future directions. The source code and datasets used in the experiments are available at Github.

Index Terms—Network Embedding, Role Discovery, Structural Similarity, Unsupervised Learning, Experimental Analysis.

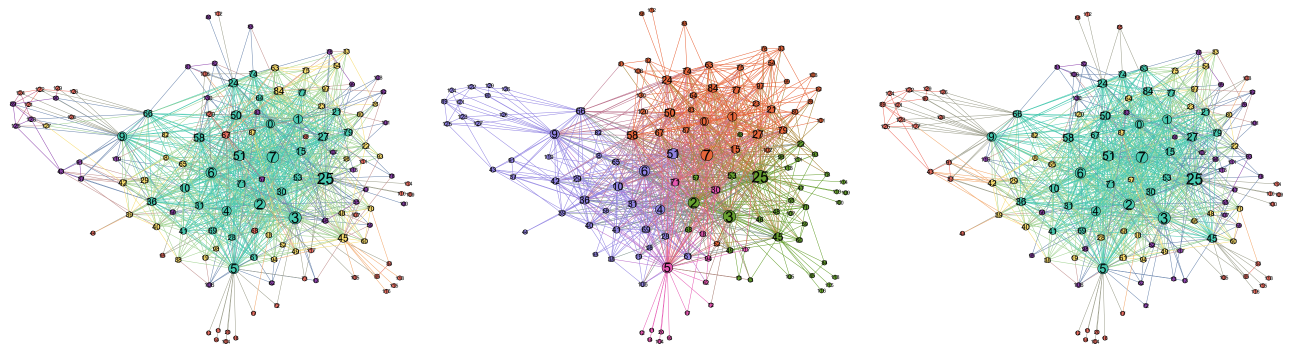
1 Introduction

NETWORK or graph is usually used to model the complex interaction relations in real-world data and systems [1], [2], e.g., transportation, social pattern, cooperative behavior and metabolic phenomenon. By convention, the network (or graph) is usually abstracted as some nodes and their complicated and elusive links. To understand such data, network analysis can help to explore the organization, analyze the structure, predict the missing links and control the dynamics in complex systems. For a long time, researchers have proposed specially designed methods and models for different graph mining tasks, such as preference mechanism, hierarchical structure and latent space model for link predication [3]; grouping or aggregation, bit compression and influence based for network summarization [4]; generalized threshold model, independent cascade model and linear Influence Model for information diffusion [5]. Among the core issues and applications of network analysis and graph mining, clustering [6], dividing the nodes into distinct or overlapping groups, has attracted attracted the most interest from different domains including machine learning and complex networks.

The field of network clustering has two main branches: **community detection** [7], [8], [9] and **role discovery** [10]. Community detection, the currently dominant clustering branch, is devoted to find common groups in which nodes interact more intensively than outside [11]. However, role discovery, which has a long research history in sociology [12] but had been inconspicuous in network science, groups the nodes based on the similarity of their structural patterns [13], such as the bridge or hub nodes [14]. In general, nodes in the same community are likely to be connected to each other, while nodes in the same role may be unconnected and often far away form each other. Since their rules on dividing nodes are fundamentally different, the two branches are usually considered as orthogonal problems. A variety of algorithms and models are proposed for both of the two branches. For community detection, the modularity optimization [15], [16], statistical model [17], [18], non-negative matrix factorization [19], [20], [21], [22] and deep learning methods [23], [24] are developed and show crucial influence for other tasks and applications, such as recommendation [25], [26] and identifying criminal gangs [27]. Some surveys on community detection can be seen in [11], [28], [29]. For role discovery, traditional methods are usually graph based and related to some equivalence, such as the structural [30], regular [31] and stochastic equivalence [32], [33]. Blockmodels [34] and mixed-membership stochastic blockmodels [35] are the important and influential methods are based on the graph. Besides, there are also some combinatorial or heuristic methods [36] for this problem.

Here we take the Brazilian air-traffic network as an example. As shown in Fig. 1 (a), the nodes and edges denote the airports and their direct flights, respectively. The clustering labels are marked based on the activity of nodes [39]. The size and color of the circle represent the degree and label of the node, respectively. It can be observed that, the nodes with the same label (color), i.e.,

Pengfei Jiao is with the College of Intelligence and Computing, Center of Biosafety Research and Strategy, Tianjin University, Tianjin, 300350, China (email: pjiao@tju.edu.cn).
Xuan Guo, Ting Pan and Wang Zhang are with the College of Intelligence and Computing, Tianjin University, Tianjin, 300350, China (email: guoxuan@tju.edu.cn; tingpan@tju.edu.cn; wangzhang@tju.edu.cn).
Yulong Pei is with the the Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, 5600MB, the Netherlands (email: y.pei.1@tue.nl).



(a) Truth label (b) Community detection (c) Role discovery

Fig. 1. Air Brazil network for understanding the community detection and role discovery in network clustering. Nodes with same clustering label have same color. (a) Brazil network with ground-truth clustering label; (b) Community detection with Louvain [37]; (c) Role discovery with RolX [38].

they are structurally similar, are usually not connected. To better illustrate these two clustering tasks, we choose two typical methods, Louvain [37] and RolX [38], which are specially designed for community detection and role discovery, respectively. The clustering results are presented in Fig. 1 (b) and (c). Community detection divides this network into some tightly connected groups, which means the airports with more flights among them belong to the same community. However, the clustering result detected by the role discovery, usually related to the flow and scale of the airport, is closer to the truth.

In recent years, network embedding (NE) has become has become the focus of studying graph structure and been demonstrated to achieve promising performance in many downstream tasks, e.g., node classification and link prediction. The motivation of NE is to transform the network data into independently distributed representations in a latent space and these representations are capable of preserving the topological structure and properties of the original network. On the whole, current methods for NE can be categorized into two types: shallow and deep learning (here we focus on unsupervised NE approaches without explicit mentioning). The former includes the matrix factorization and random walk based methods. The goal of matrix factorization methods is to learn node embedding via the low-rank approximation and approaching the adjacency matrix or higher-order similarity of the network, such as the singular value decomposition, non-negative matrix factorization and NetSMF [40]. With the different random walk strategies, a series of methods have been proposed to optimize the co-occurrence probability of nodes and learn effective embeddings. The later is mainly rooted in autoencoder and graph convolutional networks. These methods generally consist of the encoder, similarity function and decoder. There are also some attributes, characteristics and constraints that can be combined to enhance the embedding, and VGAE, GAT and GraphSAGE are some representatives of such methods. There are also some other types of embedding approaches, e.g., the latent feature model, but discussion on general NE methods is out of our scope. We refer the interested readers to some recent survey papers on NE [41], [42], [43], [44], [45], [46], [47], [48].

However, most of these methods, whether or not for network clustering, are designed for modeling the **proximity**, i.e. the embedding vectors are community oriented. They fail to capture the structural similarity, or the role information [49], Therefore, it raises several inherent challenges in the research of role-oriented network embedding. Firstly, the most and important is that two nodes with structural similarity have nothing to do with their distance, which makes it difficult to define the loss function effectively. Secondly, strict role definitions, such as some definitions based on equivalence, are difficult to be implemented in real-world networks especially large-scale networks. Thirdly, the distribution of nodes with same role in the network is very complex and interaction patterns between different roles are unknown.

In essence, there are still some scattered methods being proposed one after another recently years. These methods uses various embedding mechanisms. Struc2vec [39] leverages random walks on graphs in which edges are weighted based on structural distances. DRNE develops a deep learning framework with layer-normalized [50] LSTM model to learn regular equivalence. REACT [51], generating embeddings via matrix factorization, focus on capturing both community and role properties. Though the number of diverse role-oriented embedding methods is gradually increasing, there is still a lack of systematic understanding of role-oriented network embedding. Besides, we also lack a taxonomy for deep thinking of this problem. Meanwhile, there is short of performance and efficiency comparison of currently methods. All these limit the development and applications of role embedding.

So in this survey, we systematically analyze role-oriented network embedding and the analysis can help to understand the internal mechanism of currently methods. First, we propose a two-level categorization scheme for existing methods which is based on embedding mechanism of currently methods and models. Further more, we evaluate selected embedding methods from the perspectives of both efficiency and effectiveness on different tasks related to role discovery. In specific, we conduct comprehensive experiments on some representative methods on running time (efficiency), node classification and clustering (role discovery), top-k similarity search and visualization with widely used benchmark networks. Last, we summarize the applications, challenges and future directions of role-oriented network embedding.

Some surveys on network embedding, community detection, role discovery, and deep learning on graph have been conducted. Our survey has several essential differences compared to these works. [11], [28], [29] mainly study the problem of community detection with different focuses from the perspective of network analysis and machine learning. [10] is a seminal work in reviewing the development and methodology of role discovery. However, this survey is relatively outdated where more advanced methods, e.g., deep learning based methods, have not been discussed. Besides, these surveys focus on the methods specific for community or role task, while our work studies roles with a focus on network embedding approaches which can preserve the role information. The surveys [41], [42], [47], [48] are influential works on network embedding from different principles. However, they all focus on community-oriented methodology. Similarly, some graph embedding[1] reviews [45], [46], however, except for some technologies, have nothing to do with the role-oriented embedding. Meanwhile, surveys such as [44] and [43] introduce the effective deep learning framework and methods on graph or networks. They focus on general problems of how to use machine learning on networks and are less relevant to our problem. One relevant work is [49], it clarifies the difference between the community-oriented and role-oriented network embedding for the first time, and proposes the normal mechanisms which can help to understand if a method is designed for community or role. However, it does not systematically discuss the series of role-oriented NE methods: some advanced methods have been ignored, some introduced works are not used for role discovery or role related tasks. Moreover, it does not evaluate methods empirically by analyzing the relevant data, tasks and performance. Another recent work [52], which introduces some structural node embedding methods and evaluate them empirically, is the most relevant to our work. In analysis, they mainly focus on analyzing the relationships between NE methods and equivalence. In evaluation, they evaluate the discovered roles on direct tasks such as role classification and clustering. In contrast, we concentrate on analyzing advantages and disadvantages of different role-oriented approaches using a new two-level categorization from the analysis perspective. We conduct more comprehensive experiments to evaluate different methods w.r.t. both efficiency and effectiveness in role discovery and downstream tasks including running time, classification, clustering, visualization, and top-k similarity search.

To sum up, our survey has several contributions as follows.

- We first show the summary of role-oriented network embedding and discuss the relationship and differences of it and community oriented.
- We propose a two levels categorization schema of currently role-oriented embedding methods and briefly describe their formalization, mechanism, task, connection and difference.
- We provide full experiments of popular methods of each type on different role-oriented tasks and detailed comparison on effectiveness and efficiency.
- We share all the open-source code and widely used network datasets on Github and point out the development and questions of role-oriented network embedding.

1. We do not distinguish the difference between network embedding and graph embedding.

TABLE 1
Main Notations.

Notation	Definition
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	the network/graph with node set $\mathcal{V}$ and edge set $\mathcal{E}$
$\mathcal{N}_i^k$	the set of v_i's k-hop reachable neighbors
$\mathcal{G}_i^k$	the subgraph induced by v_i and $\mathcal{N}_i^k$
d_i	the degree of node v_i
s_{ij}	the shortest path between v_i and v_j
$\mathbf{X}$	attribute matrix
$\mathbf{I}$	identity matrix
$\mathbf{H}$	embedding matrix
$\mathbf{F}_{\mathrm{m}}$	the feature matrix extracted by or in method m
$\mathbf{S}_{\mathrm{m}}$	the similarity matrix obtained by or in method m
$\circ$, $\langle\langle\cdot\rangle\rangle$	the concatenation operator

*For conveniece, method notation m is omitted in some descriptions.

2 Notations and Framework

In this section, we give formal definitions of basic graph concepts and role-oriented network embeddings. In Table 1, we summarize the main notations used throughout this paper. Then, we propose a unified framework for understanding the process of role oriented network embedding.

Definition 1 (Network). A network is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, ..., v_n\}$ is the set of n nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges. An edge $e_{ij} = (v_i, v_j) \in \mathcal{E}$ denotes the link between node v_i and v_j.

In usual, a network $\mathcal{G}$ is represented by an weight matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$. If $e_{ij} \in \mathcal{E}$, $\mathbf{A}_{ij} > 0$ ($\mathbf{A}_{ij} = 1$ for an unweighted network and $\mathbf{A}_{ij} = \mathbf{A}_{ji}$ for an undirected network), otherwise $\mathbf{A}_{ij} = 0$. Some networks may have an attribute matrix $\mathbf{X} \in \mathbb{R}^{n\times x}$ whose ith row represents attributes of v_i. For an undirected network, denote the degree of node v_i as $d_i = \sum_j \mathbf{A}_{ij}$, and we have the degree matrix $\mathbf{D} = \mathrm{Diag}(d_1, ..., d_n)$. $\mathbf{L} = \mathbf{D} - \mathbf{A}$ is called the Laplacian matrix, it can be decomposed as $\mathbf{L} = \mathbf{U}\Lambda\mathbf{U}^\top$ where $\Lambda = \mathrm{Diag}(\lambda_1, ..., \lambda_n)$ is the matrix of eigenvalues satisfying $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$.

Denote the k-hop ($k > 0$) reachable neighbor set of node v_i as $\mathcal{N}_i^k$ (k is omitted when $k = 1$), where the shortest path between v_i and each node $v_j \in \mathcal{N}_i^k$ is less than or equal to k. For a directed network, use d_i^+, d_i^-, $\mathcal{N}_i^{k+}$ and $\mathcal{N}_i^{k-}$ to represent the out/in-degree and k-hop reachable out/in-neighborhood of v_i respectively. Unless otherwise stated, a model is discussed on unweighted undirected networks without attributes in later part of this paper.

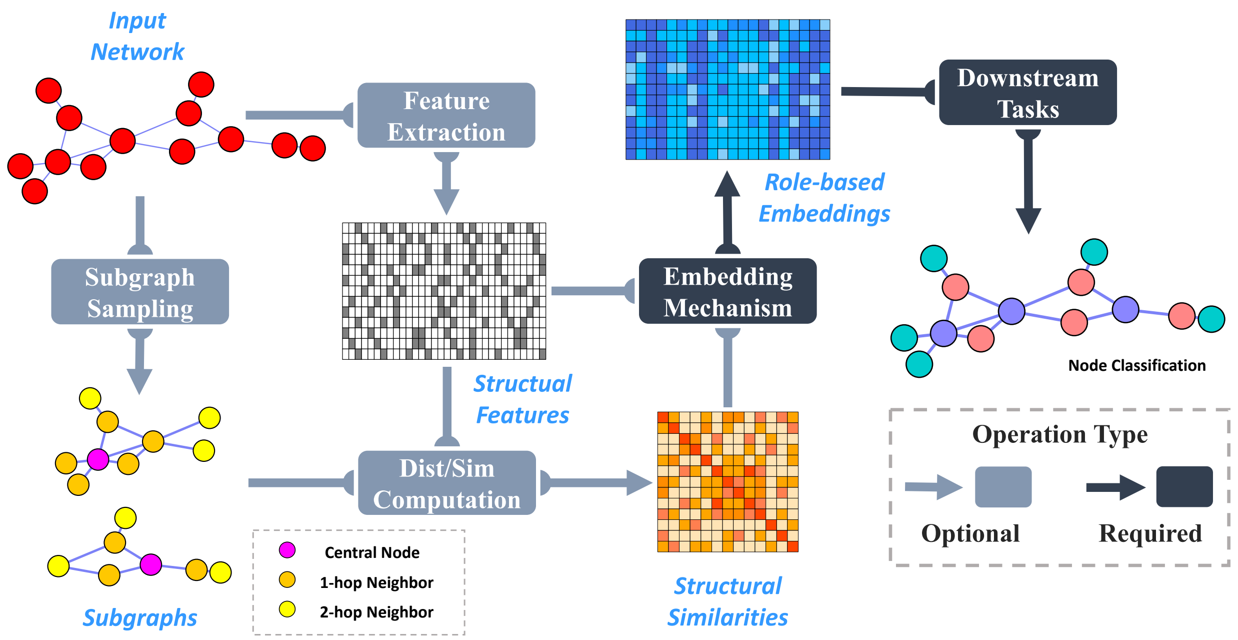


Fig. 2. The common framework of role-oriented network embedding methods includes two main step: structural property extraction and embedding. The former one can be accomplished via a variety of ways of which some are feature-based and some are similarity-based methods. On the extracted properties, role-oriented embedding methods then employ some specific embedding mechanisms to generate embeddings. Note that the discussed role-oriented methods are unsupervised. Thus, though the generated embeddings can be applied on some downstream tasks with ground truth, the whole process of embedding generation has no interaction with the target tasks.

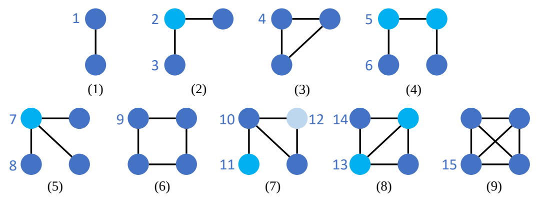


Fig. 3. Motifs and orbits (denoted by numbers) with size 2-4 nodes.

Definition 2 (Motif/Graphlet). A motif/graphlet $\mathcal{M} = (\mathcal{V}_{\mathcal{M}}, \mathcal{E}_{\mathcal{M}})$ is a small connected subgraph representing particular patterns of edges on several nodes. The pattern can be repeated in or across networks, i.e., many subgraphs can be sampled from networks and isomorphic to it. Nodes automorphic to each other, i.e., having the same connectivity patterns, are in the same orbits.

For unweighted networks, there are 9 motifs and 15 orbits with size 2-4 nodes as shown in Fig. 3. Because of their ability to model the smallest but most fundamental connectivity patterns, motifs are wildly used for capturing structural similarities and discovering roles.

Definition 3 (Network Clustering). A clustering $\boldsymbol{\mathcal{R}} = \{\mathcal{R}_1, ..., \mathcal{R}_k\}$ of network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a group of node sets satisfying $\forall 1 \le i \le k : \mathcal{R}_i \neq \emptyset, \mathcal{R}_i \subset \mathcal{V}$ and $\bigcup_1^k \mathcal{R}_i = \mathcal{V}$. In this paper, we discuss about hard clustering, i.e., $\forall 1 \le i < j \le k : \mathcal{R}_i \cap \mathcal{R}_j = \emptyset$. If $\mathcal{R}_i \cup \mathcal{R}_j \neq \emptyset$, it is usually called overlapping or soft clustering.

For the **community detection**, each set $\mathcal{R}_i$ is a tightly interconnected collection of nodes. And for **role discovery**, it usually composed of unconnected nodes which have similar structural patterns or functions. So every network clustering algorithm is committed to achieve the clustering results under different goal constraints. However, there is no common understanding of role equivalence or similarity, which leads to multifarious definitions of equivalence and designs of similarity computation. For example, two nodes are automorphic equivalent [53] as the subgraphs of their neighborhood are isomorphic, while regular equivalence [54] means that if two nodes have the same roles, there neighbors have the same roles.

Definition 4 (Network Embedding). Network Embedding is a process to map nodes of network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to low-dimensional embeddings $\mathbf{H} \in \mathbb{R}^{n \times r}$ so that $r \ll n$. In general, for nodes v_i and v_j, if they are similar in the network, their embedding vectors $\mathbf{H}_i$ and $\mathbf{H}_j$ will be close in the low dimensional space.

With the node embedding, we can take it for different network tasks. If we focus on the community detection or link predication, we want to discriminate by embeddings whether the nodes are connected or likely to be connected. However, for role discovery, the embeddings should reflect some structural patterns including local properties like subgraph isomorphism, global properties like regular equivalence and higher-order properties like motifs.

Based on the discussion and notations above, here we firstly propose a unified framework for understanding the role-oriented network embedding. To our knowledge, it can cover almost all the existing methods and models in a unified way. The framework is illustrated in Fig. 2. Structure of networks is discrete, but embeddings are usually designed to lie in continuous space. Thus, role-oriented embedding methods always take two steps to capture structrual properties and generate embeddings respectively to bridge the gulf between two spaces:

- **Structure Property Extraction**. The ways to extract structural information are diverse. Some methods such as RolX [38] and DRNE [50] leverages some primary structural features including node degree, triangle numbers. Part of these methods such as SPINE [55] will continue to transform the features into distances or similarities. There are also some methods captureing similarity between node-centric subgraphs. For example, struc2vec [39] compute structural distances between k-hop subgraphs based on degree sequences of the subgraphs. SEGK [56] employs one graph isomorphism test skill called graph kernel on subgraphs. As the result of these hand-craft process, these structural properties are contained in interim features or pair-wise similarities.
- **Embedding**. The extracted properties then are mapped into embedding space via different mechanisms. In the process of embedding, the structural properties are used as inputs or training guidance. For example, RolX [38] and SEGK [56] apply low-rank matrix factorization on feature matrix and similarity matrix respectively, as they implicitly or explicitly reflect whether nodes are structurally similar. Struc2vec [39], [55] leverages word embedding methods to the similarity-biased random walks. DRNE [50] utilizes LSTM [57] on degree-ordered node embedding sequences to capture regular equivalence with a degree-guided regularizer.

As the embeddings capture crucial structural properties, they can be used on the downstream tasks such as role-based node classification and visualization. With this framework, we generalize the process of role-oriented embedding. However, as we can learn from Fig. 2, the core of designing role-oriented embedding methods is the way to extract structural properties. In contrast, the embedding mechanisms for mapping structural features/similarities into low-dimensional continuous vector space are much more regular. Thus, We introduce the popular methods in the next section from the perspective of embedding mechanisms.

3 Algorithm Taxonomy

In this section, we introduce these approaches categorized according to their embedding mechanisms. In detail, we propose a two-level classification ontology for these popular methods. Similar to the taxonomy of community oriented network embedding, we divide these into three categories, low-rank matrix factorization, random walk based and deep learning methods from the first level. Further more, with there embedding mechanisms and constraint information, we give a more refined classification taxonomy as shown in TABLE 2. At the same time, we also list the tasks which can be served by different methods. Next, we will introduce these methods in detail.

3.1 Low-rank Matrix Factorization

Low-rank matrix factorization is the most commonly used method for role-oriented embeddding methods. They generate embeddings by factorizing matrices preserving the role similarities between nodes implicitly (i.e. feature matrices) or explicitly (i.e. similarity matrices).

3.1.1 Structural Feature Matrix Factorization

RolX [38]. RolX takes the advantages of feature extraction method ReFeX [72] by decomposing the ReFeX feature matrix $\mathbf{F}_{ReFeX} \in \mathbb{R}^{n\times f}$. ReFeX firstly computes some primary features such as degree and clustering coefficient for each node. Then it aggregates neighbors' features with sum- and mean-aggregator recursively. In k recursive steps, it can capture very thorough features to express the structure of $(k+1)$-hop reachable neighborhood. Non-negative Matrix Factorization (NMF) is used for generating embeddings as it is efficient compared with other matrix decomposition methods. The non-negative constraints are adapted to interpretation of roles. Thus, RolX aims to obtain two low-rank matrices as follows:

$$\min_{\mathbf{H},\mathbf{M}} \left|\left|\mathbf{F}_{ReFeX} - \mathbf{HM}\right|\right|_F^2,\ s.t.\ \mathbf{H},\mathbf{M} \geq 0 \quad (1)$$

where $\mathbf{H} \in \mathbb{R}^{n\times r}$ is the embedding matrix (or role assignment matrix) and the matrix $\mathbf{M} \in \mathbb{R}^{r\times f}$ (role definition matrix) describes the contributions of each role to structural features. r is the number of hidden roles which is determined by Minimum Description Length (MDL) [73].

GLRD [14]. GLRD extends RolX by adding different optional constraints to objective function (1). Sparsity constraint ($\forall i, \|\mathbf{H}_{\cdot i}\|_1 \leq \epsilon_{\mathbf{H}} \wedge \|\mathbf{M}_{i\cdot}\|_1 \leq \epsilon_{\mathbf{M}}$) is defined for more definitive role assignments and definitions while diversity constraint ($\forall i \neq j, \mathbf{H}_{\cdot i}^\top \mathbf{H}_{\cdot j} \leq \epsilon_{\mathbf{H}} \wedge \mathbf{M}_{i\cdot}\mathbf{M}_{j\cdot}^\top \leq \epsilon_{\mathbf{M}}$) is for reducing the overlapping. $\mathbf{H}^*$ and $\mathbf{M}^*$ are previously discovered role assignments and definitions with which alternativeness constraint ($\forall i \neq j, \mathbf{H}_{\cdot i}^{*\top}\mathbf{H}_{\cdot j} \leq \epsilon_{\mathbf{H}} \wedge \mathbf{M}_{i\cdot}^*\mathbf{M}_{j\cdot}^\top \leq \epsilon_{\mathbf{M}}$) can be used for mining roles unknown.

RIDεRs [58]. RIDεRs uses ε-equitable refinement (εER) to partition nodes into different cells and compute graph-based features. An ε-equitable refinement partition $\pi = \{\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_K\}$ of $\mathcal{V}$ satisfies the following rule:

$$|\deg(u,\mathcal{C}_j) - \deg(v,\mathcal{C}_j)| \leq \varepsilon, \forall u,v \in \mathcal{C}_j, \forall 1 \leq i,j \leq K \quad (2)$$

where $\deg(u,\mathcal{C}_j) = |\{u|(u,v) \in E \wedge v \in \mathcal{C}_j\}|$ denotes the number of nodes in cell $\mathcal{C}_j$ connected to node v_i. As nodes in the same cell have similar number of connections to the nodes in another cell, εERs could capture some connectivity patterns.

Based on the cells partitioned by εERs with an relaxation parameter ε, the feature matrix is defined as $(\mathbf{F}^{\varepsilon}_{\varepsilon ER})_{ij} = |\mathcal{N}_i \cap \mathcal{C}_j|$. After prunning and binning process, the feature matrices for all $1 \leq \varepsilon \leq \lfloor d_{avg} \rfloor$ are concatenated as the final feature matrix $\mathbf{F}_{\varepsilon ER}$. Finally, like RolX and GLRD, NMF is applied for embedding generation while right sparsity constraint (on $\mathbf{M}$) is optional for more definitive role representations.

GraphWave [59]. GraphWave treats graph diffusion kernels as probability distributions over networks and gets embeddings by using characteristic functions of the distributions.

TABLE 2
A summary of role-oriented embedding methods. The abbreviations of tasks CLF, CLT, LP, ER, NA, SS and Vis denote node classification/clustering, link prediction, entity resolution/Network alignment/Top-k similarity search and visualization, respectively.

Method	Embedding Mechanism		Conducted Tasks				Year
			Vis	CLF/CLT	ER/NA/SS	LP	
RolX [38]	low-rank matrix factorization (Sec.3.1)	on structural feature matrix (Sec.3.1.1)	✓	✓	✗	✗	2012
GLRD [14]			✗	✗	✓	✗	2013
RIDεRs [58]			✓	✓	✓	✗	2017
GraphWave [59]			✓	✓	✗	✗	2018
HONE [60]			✓	✗	✓	✓	2020
xNetMF [61]		on structural similarity matrix (Sec.3.1.2)	✗	✗	✓	✗	2018
EMBER [62]			✗	✓	✓	✗	2019
SEGK [56]			✓	✓	✓	✗	2019
REACT [51]			✗	✓	✗	✗	2019
SPaE [63]			✓	✓	✗	✗	2019
struc2vec [39]	random walk-based methods (Sec.3.2)	on similarity-biased random walks (Sec.3.2.1)	✓	✓	✗	✗	2017
SPINE [55]			✗	✓	✗	✗	2019
struc2gauss [64]			✓	✓	✗	✗	2020
Role2Vec [65]		on feature-based random walks (Sec.3.2.2)	✗	✗	✗	✓	2019
RiWalk [66]			✗	✓	✗	✗	2019
NODE2BITS [67]			✗	✗	✓	✗	2019
DRNE [50]	deep learning (Sec.3.3)	via structural information reconstruction/guidance (Sec.3.3.1)	✓	✓	✗	✗	2018
GAS [68]			✓	✓	✗	✗	2020
RESD [69]			✓	✓	✓	✗	2021
GraLSP [70]			✓	✓	✗	✓	2020
GCC [71]			✗	✓	✓	✗	2020

Specifically, take the heat kernel $g_\varsigma(\lambda) = e^{-\lambda\varsigma}$ with scaling parameter ς as an example, the spectral graph wavelets $\mathbf{\Psi} \in \mathbb{R}^{n\times n}$ are defined as:

$$\mathbf{\Psi} = \mathcal{I}\mathbf{U}\text{Diag}(g_\varsigma(\lambda_1), ..., g_\varsigma(\lambda_n))\mathbf{U}^\top \tag{3}$$

where $\mathcal{I}$ is the one-hot encoding matrix on $\mathcal{V}$ and the scaling parameter ς is omitted. The i-th row $\mathbf{\Psi}_i$ represents the resulting signal from a Dirac signal around node v_i. Considering the empirical characteristic function:

$$\varphi_i(t) = \frac{1}{n}\sum_{j=1}^{n} e^{\mathrm{i}t\mathbf{\Psi}_{ij}} \tag{4}$$

where i denotes the imaginary number, v_i's embedding vector $\mathbf{H}_i$ is generated by concatenating pairs of $\text{Re}(\varphi_i(t))$ and $\text{Im}(\varphi_i(t))$ at r evenly spaced points $t_1, ..., t_r$.

HONE [60]. HONE constructs weighted motif graphs in which the weight of an edge is the count of the co-occurrences of the two endpoints in a specific motif. For a motif represented by its weighted motif adjacency matrix $\mathbf{A}_{\mathcal{M}_m}$, HONE characters the higher-order structure by deriving matrices from its k-step matrices $\mathbf{A}^k_{\mathcal{M}_m}$. These new matrices are designed by imitating some popular matrices based on normal adjacency matrix such as transition matrix $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$ and Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$. Here we use $(\mathbf{F}_{HONE})^{(k)}_{\mathcal{M}_m}$ to denote the derived matrices. Then the k-step embeddings can be learned as:

$$\underset{\mathbf{H}^{(k)}_{\mathcal{M}_m}, \mathbf{M}^{(k)}_{\mathcal{M}_m}}{\arg\min} \ \mathbb{D}_{Breg}(\mathbf{F}^{(k)}_m | \Psi(\mathbf{H}^{(k)}_{\mathcal{M}_m}\mathbf{M}^{(k)}_{\mathcal{M}_m})) \tag{5}$$

where $\mathbb{D}_{Breg}$ is the Bregman divergence and $\Psi(\cdot)$ is a matching function. The global embeddings are generated by minimizing the following objective:

$$\min_{\mathbf{H},\mathbf{M}} \left|\left|\mathbf{F}_{HONE} - \mathbf{HM}\right|\right|^2_F \tag{6}$$

where $\mathbf{F}_{HONE}$ is obtained by concatenating the $\mathbf{H}^{(k)}_{\mathcal{M}_m}$ with all the considered motifs and steps. If necessary, attributes diffused by transition matrix based on different motifs and steps can be added into $\mathbf{F}_{HONE}$.

Remark. Aforementioned methods assume that nodes in similar roles have similar structural features. Thus, they apply matrix factorization on the feature matrices to obtain role-based representations. RolX, GLRD and RIDεRs directly get embeddings which give soft role assignment by factorizing feature matrices. As the feature matrices are usually lower dimension, these methods are quite efficient. GraphWave uses eigen-decomposition and empirical characteristic function to characterize the structural patterns of each node, which leads to robust embeddings but high computation cost. The weighted motif adjacency matrices in HONE capture higher-order proximities actually, while they can obtain structural information because each matrix represents one motif.

3.1.2 *Structural Similarity Matrix Factorizaiton*

xNetMF [61]. xNetMF is an embedding method designed for an embedding-based network alignment approach REGAL. It firstly obtains a node-to-node similarity matrix $\mathbf{S}_{REGAL}$ based on both structures and attributes:

$$\mathbf{S}_{ij} = \exp(-\gamma_s \text{dist}_s(v_i, v_j) - \gamma_a \text{dist}_a(v_i, v_j)) \tag{7}$$

where $\mathrm{dist}_s(v_i, v_j)$ and $\mathrm{dist}_a(v_i, v_j)$ are structure-based and attribute-based distance between node v_i and node v_j while γ_s and γ_a are balance parameters of the two distances. $\mathrm{dist}_s(v_i, v_j)$ is the Euclidean distance between node features. And $\mathrm{dist}_a(v_i, v_j)$ counts different attributes between nodes, i.e., $\mathrm{dist}_a(v_i, v_j) = |\{a|\mathbf{X}_{ia} \neq \mathbf{X}_{ja} \wedge 1 \leq a \leq x\}|$. The feature matrix $\mathbf{F}_{REGAL}$ is defined by counting nodes with the same logarithmically binned degree in each node's k-hop reachable neighborhood as follows:

$$\begin{aligned} \mathbf{F}_{ic}^k &= |\mathcal{D}_{i,c}^k| = |\{v_j \in \mathcal{N}_i^k | \lfloor \log_2 d_j \rfloor = c\}|, \\ \mathbf{F}_i &= \sum_{k=1}^{K} \delta^k \mathbf{F}_i^k \end{aligned} \tag{8}$$

where $\delta^k \in (0, 1]$ is a discount factor for lessening the importance of higher-hop neighbors.

Then on computed $\mathbf{S}$, matrix factorization methods can be applied for obtaining embedding matrix $\mathbf{H}$ satisfying $\mathbf{S} \approx \mathbf{H}\mathbf{M}^\top$. As the high dimension and rank of $\mathbf{S}$ lead to high computation, an implicit matrix factorization approach extending Nyström method [74] is proposed as follows:

1) Select $r \ll n$ nodes as landmarks randomly or based on node centralities.
2) Compute a node-to-landmark similarity matrix $\mathbf{C} \in \mathbb{R}^{n \times r}$ with Eq.(7) and extract a landmark-to-landmark similarity matrix $\mathbf{B} \in \mathbb{R}^{r \times r}$ from $\mathbf{C}$.
3) Apply Singular Value Decomposition on the pseudoinverse of $\mathbf{B}$ so that $\mathbf{B}^\dagger = \mathbf{V}\mathbf{\Sigma}\mathbf{Y}^\top$.
4) Obtain embedding matrix $\mathbf{H}$ by computing and normailize $\mathbf{C}\mathbf{V}\mathbf{\Sigma}^{-\frac{1}{2}}$.

With above method, embeddings are actually generated by factorizing a low-rank approximation of $\mathbf{S}$, i.e., $\widetilde{\mathbf{S}} = \mathbf{C}(\mathbf{V}\mathbf{\Sigma}\mathbf{Y}^\top)\mathbf{C}^\top$. Meanwhile, the computation can be reduced, as only a small matrix $\mathbf{B}^\dagger$ is decomposed.

EMBER [62]. EMBER is designed for mining professional roles in weighted directed email networks. It defines node outgoing feature matrix $\mathbf{F}_{EMBER}^+$ as:

$$\begin{aligned} \mathbf{F}_{ic}^{k+} &= \sum_{v_j \in \mathcal{D}_{i,c}^{k+}} \mathrm{pw}(\mathcal{P}_{v_i \to v_j}^{k+}), \\ \mathbf{F}_i^+ &= \sum_{k=1}^{K} \delta^k \mathbf{F}_i^{k+} \end{aligned} \tag{9}$$

where $\mathrm{pw}(\mathcal{P}_{v_i \to v_j}^{k+})$ denotes the product of all edge weights in a k-step shortest outgoing path $\mathcal{P}_{v_i \to v_j}^{k+}$. The incoming feature matrix $\mathbf{F}$ is defined similarly. By concatenating the incoming and outgoing feature matrices, the final feature matrix $\mathbf{F}_{EMBER} = [\mathbf{F}^+, \mathbf{F}^-]$ is obtained. The node-to-node similarities are computed through Eq.(7) without attribute-based distance, i.e., $\mathbf{S}_{ij} = \exp(-\|\mathbf{F}_i - \mathbf{F}_j\|^2)$. EMBER uses the same implicit matrix factorization approach to generate embeddings. Note that if the feature extraction part of EMBER is applied on an undirected unweight network, EMBER will be equivalent to xNetMF without attributes.

SEGK [56]. SEGK leverages graph kernels to compute node structural similarities. To compare the structure more carefully, it computes node similarities with different scales of neighborhood as follows:

$$\mathbf{S}_{ij} = \sum_{k=1}^{K} \hat{\mathcal{K}}(\mathcal{G}_i^k, \mathcal{G}_j^k) \hat{\mathcal{K}}(\mathcal{G}_i^{k-1}, \mathcal{G}_j^{k-1}) \tag{10}$$

where $\hat{\mathcal{K}}(\mathcal{G}_i^0, \mathcal{G}_j^0) = 1$ and $\hat{\mathcal{K}}$ denotes the normalized kernel which is defined as:

$$\hat{\mathcal{K}}(\mathcal{G}, \mathcal{G}') = \frac{\mathcal{K}(\mathcal{G}, \mathcal{G}')}{\sqrt{\mathcal{K}(\mathcal{G}, \mathcal{G})\mathcal{K}(\mathcal{G}', \mathcal{G}')}} \tag{11}$$

SEGK chooses the shortest path kernel, Weisfeiler-Lehman subtree kernel, or graphlet kernel for practical use of $\mathcal{K}(\cdot, \cdot)$. Then Nyström method [75] is employed on the factorization of $\mathbf{S}$ for efficient computation and low dimensions of embeddings as follows:

$$\mathbf{H} = \mathbf{S}\mathbf{U}_{[r]}\mathbf{\Lambda}_{[r]}^{-\frac{1}{2}} \tag{12}$$

where $\mathbf{U}_{[r]}$ denotes the matrix of first r eigenvectors and $\mathbf{\Lambda}_{[r]}$ is the diagonal matrix of corresponding eigenvalues.

REACT [51]. REACT aims to detect communities and discover roles by applying non-negative matrix tri-factorization on RoleSim [76] matrix $\mathbf{S}$ and adjacency matrix $\mathbf{A}$, respectively. RoleSim matrix is developed with the idea of regular equivalence and is a pair-wise similarity matrix computed by iteratively updating the following scores:

$$\mathbf{S}_{ij} = (1-\beta) \max_{\mathrm{M}(v_i, v_j)} \frac{\sum_{(v_{i'}, v_{j'}) \in \mathrm{M}(v_i, v_j)} \mathbf{S}_{i'j'}}{d_i + d_j - |\mathrm{M}(v_i, v_j)|} + \beta \tag{13}$$

where $\mathrm{M}(v_i, v_j)$ is a matching between the neighborhoods of v_i and v_j, and β ($0 < \beta < 1$) is a decay factor. In addition, $L_{2,1}$ norm is leveraged as the regularization to make the distribution of roles within communities as diverse as possible. Thus, the objective function of REACT is:

$$\begin{aligned} \min_{\mathbf{H}_R, \mathbf{M}_R, \mathbf{H}_C, \mathbf{M}_C} & \left\| \mathbf{S} - \mathbf{H}_R \mathbf{M}_R \mathbf{H}_R^\top \right\|_F^2 \\ & + \left\| \mathbf{A} - \mathbf{H}_C \mathbf{M}_C \mathbf{H}_C^\top \right\|_F^2 + \gamma_{2,1} \left\| \mathbf{H}_C^\top \mathbf{H}_R \right\|_{2,1}, \\ s.t. \quad & \mathbf{H}_R, \mathbf{M}_R, \mathbf{H}_C, \mathbf{M}_C \geq 0, \mathbf{H}_R^\top \mathbf{H}_R = \mathbf{I}, \mathbf{H}_C^\top \mathbf{H}_C = \mathbf{I}. \end{aligned} \tag{14}$$

where $\mathbf{H}_R/\mathbf{H}_C$ denotes the embedding matrix for roles/communities, and $\mathbf{M}_R/\mathbf{M}_C$ denotes the interaction between roles/communities. $\gamma_{2,1}$ is the weight of regularization. Orthogonal constraint on embedding matrices is added for increased interpretability.

SPaE [51]. SPaE also tries to capture communities and roles simultaneously. For node structural similarity, it computes cosine similarity between the standardized Graphlet Degree Vectors of nodes, and generates role-based embeddings via Laplacian eigenmaps method as follows:

$$\max_{\mathbf{H}_R} \mathcal{J}_R = \mathrm{Tr}(\mathbf{H}_R^\top \mathbf{L}_\mathbf{S} \mathbf{H}_R), \; s.t.\; \mathbf{H}_R^\top \mathbf{H}_R = \mathbf{I}. \tag{15}$$

where $\mathbf{L}_\mathbf{S}$ is the symmetric normalized matrix of structural similarity matrix $\mathbf{S}_{SPaE}$. SPaE obtains community-based embeddings similarly as follows:

$$\max_{\mathbf{H}_C} \mathcal{J}_C = \mathrm{Tr}(\mathbf{H}_C^\top \mathbf{L}_\mathbf{A} \mathbf{H}_C), \; s.t.\; \mathbf{H}_C^\top \mathbf{H}_C = \mathbf{I}. \tag{16}$$

where $\mathbf{L_A} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ is the symmetric normalized adjacency matrix. To map $\mathbf{H}_R$ and $\mathbf{H}_C$ into a unified embedding space, SPaE generates hybrid embeddings by maximizing the following objective function:

$$\begin{aligned} &\max_{\mathbf{H}_R,\mathbf{H}_C,\mathbf{H}_H} \mathcal{J}_R + \mathrm{p}_R + \gamma(\mathcal{J}_C + \mathrm{p}_C), \\ &s.t.\ \mathbf{H}_R^\top\mathbf{H}_R = \mathbf{I}, \mathbf{H}_C^\top\mathbf{H}_C = \mathbf{I}, \mathbf{H}_H^\top\mathbf{H}_H = \mathbf{I}. \end{aligned} \tag{17}$$

where $\mathbf{H}_H$ denotes the hybrid embedding matrix and γ is the balance parameter. $\mathrm{p}_R = \mathrm{Tr}(\mathbf{H}_R^\top\mathbf{H}_H\mathbf{H}_H^\top\mathbf{H}_R)$ and $\mathrm{p}_C = \mathrm{Tr}(\mathbf{H}_C^\top\mathbf{H}_H\mathbf{H}_H^\top\mathbf{H}_C)$.

Remark. These methods all explicitly compute structural similarities based on features, e.g., graph kernels, role equivalence, and so on. Most of them have considered the similarities between multiple hops of neighborhoods. Their effectiveness on role discovery depends on the quality of the similarity matrices. One major problem of this kind of methods is the issue of efficiency: computing pair-wise similarity and factorizing the high-dimensional similarity matrix $\mathbf{S} \in \mathbb{R}^{n\times n}$ are time-consuming. So xNetMF, EMBER and SEGK apply Nyström method to improve the efficiency as their similarity matrices are Gram matrices [74].

3.2 Shallow Models Using Random Walks

Random walk is a common way to capture node proximity used by network embedding methods [77], [78]. Recently, two strategies have been proposed to adapt random walks to role-oriented tasks: (1) structural similarity-biased random walks makes structurally similar nodes more likely to appear in the same sequence (as shown in Fig. 4(b)). (2) structural feature-based random walks, e.g., attributed random walks [65], map nodes with similar structural features to the same role indicator and replace ids in random walk sequences with the indicators (see Fig. 4(c)). The first way can preserve structural similarity into co-occurrence relations of nodes in the walks. While the second way preserves structural similarity into role indicators and may capture the proximity between roles through the co-occurrence relations of the indicators as well.

Usually, language models such as Skip-Gram [79] are applied on generated random walks to map the similarities into embedding vectors [39], [65], [77], [78]. However, some different mapping mechanisms are also employed such as the SimHash [80] used in NODE2BITS [67].

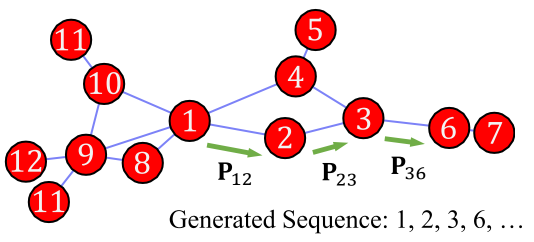
3.2.1 Structural Similarity-biased Random Walks

struc2vec [39]. Struc2vec generates structurally biased id-based contexts via random walks on a hierarchy of constructed complete graphs.

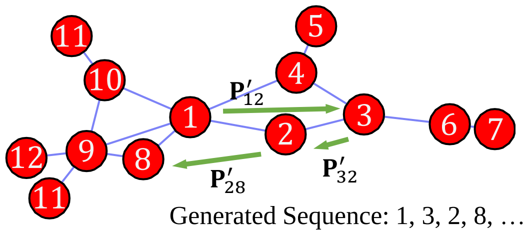
In detail, it firstly computes structural distances between a pair of nodes as follows:

$$\begin{aligned} \mathrm{dist}_d^k(v_i, v_j) =& \mathrm{dist}_d^{k-1}(v_i, v_j) + \mathrm{DTW}(\mathcal{H}_i^k, \mathcal{H}_j^k), \\ & 0 \le k \le k^* \end{aligned} \tag{18}$$

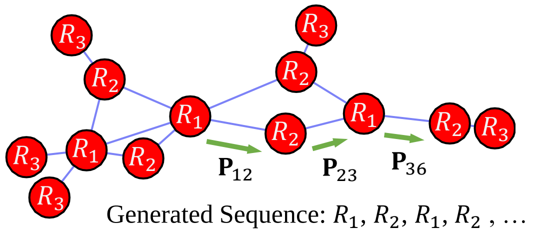
where $\mathrm{DTW}(\cdot,\cdot)$ denotes Dynamic Time Warping (DTW). $\mathrm{d}(a, b) = \max(a, b)/\min(a, b) - 1$ is adopted as the distance function for DTW. k^* is the diameter of the $\mathcal{G}$. $\mathcal{H}_i^k$ is the ordered degree sequence of nodes at the exact distance k from v_i. Note that $\mathcal{H}_i^0 = \{d_i\}$ and $\mathrm{dist}_d^{-1}(v_i, v_j)$ is set to the constant 0.



(a) Normal Random Walks



(b) Structural Similarity-biased Random Walks



(c) Structural Feature-based Random Walks

Fig. 4. Different types of random walks. Note that $\mathbf{P}'$ is the biased transition matrix computed based on node structural similarities. R_i is the role indicator mapped from node structural features.

Then a multi-layer weighted context graph $\mathcal{G}_C = (\mathcal{V}_C, \mathcal{E}_C)$ is built. Each layer $k = 0, ..., k^*$ is an undirected complete graph $\mathcal{G}_C^k = (\mathcal{V}_C^k, \mathcal{E}_C^k)$. $\mathcal{V}_C^k = \{v_1^k, ..., v_n^k\}$ where the corresponding node of $v_i \in \mathcal{V}$ in k-layer is denoted as v_i^k. The weight $w_C^k(v_i^k, v_j^k)$ of edge $(v_i^k, v_j^k) \in \mathcal{E}_C^k$ is defined as follows:

$$w_C^k(v_i^k, v_j^k) = \exp(-\mathrm{dist}_d^k(v_i^k, v_j^k)), k = 0, ..., k^* \tag{19}$$

The neighboring layers are connected through directed edges between the corresponding nodes. Thus $\mathcal{V}_C = \bigcup_0^{k^*} \mathcal{V}_C^k$ and $\mathcal{E}_C = (\bigcup_0^{k^*} \mathcal{E}_C^k) \cup (\bigcup_1^{k^*} \bigcup_1^n \{(v_i^k, v_i^{k-1})\}) \cup (\bigcup_0^{k^*-1} \bigcup_1^n \{(v_i^{k-1}, v_i^k)\})$. The edge weights between layers are defined as follows:

$$\begin{aligned} w_C(v_i^k, v_i^{k+1}) =& \log(\Gamma(v_i^k) + \mathrm{e}), k = 0, ..., k^* - 1 \\ w_C(v_i^k, v_i^{k-1}) =& 1, k = 1, ..., k^* \end{aligned} \tag{20}$$

where $\Gamma(v_i^k)$ counts the edge $(v_i^k, v_j^k) \in \mathcal{E}_C^k$ whose weight is larger than the average edge weight of $\mathcal{G}_C^k$. That is:

$$\begin{aligned} \Gamma(v_i^k) = |\{(v_i^k, v_j^k)| w_C^k(v_i^k, v_j^k) > \\ \frac{\sum_{(v_{i'}^k, v_{j'}^k)\in\mathcal{E}_C^k} w_C^k(v_{i'}^k, v_{j'}^k)}{\binom{n}{2}}\}| \end{aligned} \tag{21}$$

Then id-based random walks can be employed on $\mathcal{G}_C$ and started in layer 0 for context generation of each node.

In detail, the walk stays in the current layer with a given probability q_b. In this situation, the probability of a walk from v_i^k to v_j^k is:

$$(\mathbf{P}_{S2V}^k)_{ij} = \frac{w_C^k(v_i^k, v_j^k))}{\sum_{(v_i^k, v_{j'}^k)\in\mathcal{E}_C^k} w_C^k(v_i^k, v_{j'}^k))} \tag{22}$$

With probability $1-q_b$, the walk steps across layers with the following stepping probability:

$$\begin{aligned} p(v_i^k, v_i^{k+1}) =& \frac{w_C(v_i^k, v_i^{k+1})}{w_C(v_i^k, v_i^{k+1}) + w_C(v_i^k, v_i^{k-1})} \\ p(v_i^k, v_i^{k-1}) =& 1 - p(v_i^k, v_i^{k+1}) \end{aligned} \tag{23}$$

Note that v_i^k with different k have the same id in the context.

On the structural context, struc2vec leverages Skip-Gram with Hierarchical Softmax to learn embeddings.

SPINE [55]. SPINE uses largest f values of ith row of Rooted PageRank Matrix $\mathbf{\Omega} = (1 - \beta_{RPR})(\mathbf{I} - \beta_{RPR}\mathbf{P})^{-1}$ as v_i's feature $(\mathbf{F}_{RPR})_i$. β_{RPR} is the probability of stepping to a neighbor, while with probability $1-\beta_{RPR}$, a walk steps back to the start node. For the inductive setting, SPINE computes $\mathbf{F}_i$ via a Monte Carlo approximation.

To simultaneously capture structural similarity and proximity, SPINE designs a biased random walk method. With probability ϱ_{br}, the walk steps to a structural similar node based on the following transition matrix:

$$(\mathbf{P}_{SPINE}^k)_{ij} = \frac{\text{sim}(v_i, v_j)}{\sum_{v_k\in\mathcal{V}, v_k\neq v_i} \text{sim}(v_i, v_k)} \tag{24}$$

Here $\text{sim}(\cdot,\cdot)$ can be computed via DTW or other methods based on node features. With probability $1-\varrho_{br}$, normal random walks are applied. Thus, with larger ϱ_{br}, SPINE can be more role-oriented. The embeddings are learned through Skip-Gram with Negative Sampling (SGNS). To leverage attributes, the embeddings are generated as:

$$\mathbf{H}_i = \sigma(\sum_{j=1}^{f} \mathbf{F}_{ij}\mathbf{X}_j^{<i,f>}\mathbf{W}) \tag{25}$$

where $\mathbf{X}^{<i,f>}$ represents the attribute matrix of which the rows correspond to f largest values of $\mathbf{\Omega}_i$. $\mathbf{W}$ is the weight matrix of the multi-layer perceptron (MLP).

struc2gauss [64]. For each node v_i, struc2gauss generates a Gaussian distribution: $\mathcal{Z}_i = \text{Gauss}(\boldsymbol{\mu}_i, \mathbf{\Sigma}_i)$ to model both structural similarity and uncertainty. After calculating structural similarity via existing methods such as RoleSim [76], it samples the top-K most similar nodes for a node as its positive set $\mathcal{S}_i^+$. The positive sampling of struc2gauss could be regarded as special random walks with mandatory restart on a star-shaped graph where the star center is the target node and star edges are the most similar nodes. The negative sample set $\mathcal{S}_i^-$ has the same size of $\mathcal{S}_i^+$ and is generated as in the normal random-walk based methods. To push the Gaussian embeddings of similar nodes closer and those of dissimilar nodes farther, struc2gauss uses the following max-margin ranking objective:

$$\mathcal{L} = \sum_{v_i\in\mathcal{V}} \sum_{v_j\in\mathcal{S}_i^+} \sum_{v_k\in\mathcal{S}_i^-} \max\left(0, m - \text{sim}\left(\mathcal{Z}_i, \mathcal{Z}_j\right) + \text{sim}\left(\mathcal{Z}_i, \mathcal{Z}_k\right)\right) \tag{26}$$

where m is the margin parameter to push dissimilar distributions apart, and $\text{sim}\left(\mathcal{Z}_i, \mathcal{Z}_j\right)$ is the similarity measure between distributions of v_i and v_j. There are different similarity measures that can be used such as logarithmic inner product and KL divergence. For normal tasks, the mean vectors of those Gaussian distributions can be treated as embeddings, i.e., $\mathbf{H}_i = \boldsymbol{\mu}_i$.

Remark. These methods reconstruct the edges between nodes based on the structural similarities so that the context nodes obtained by random walks are structurally similar to the central nodes. Compared with SPINE and struc2gauss, struc2vec clearly construct edges that better represent role information in the multi-layer complete graphs, which leads to better embeddings but higher time and space complexities.

3.2.2 *Structural Feature-based Random Walks*

Role2Vec [65]. Role2Vec firstly maps nodes into several disjoint roles. Logarithmically binning, K-means with low-rank factorization and other methods on features and attributes can be chosen for the role mapping $\phi : \mathcal{V} \rightarrow \mathcal{R}$. Motif-based features, such as Graphlet Degree Vectors, are recommended since motif can better capture the high-order structural information.

Then random walks are performed but the ids in generated sequences are replaced with role indicators. With feature-based role context, the language model CBOW can be used for obtaining embeddings of roles. Nodes partitioned into the same role have the same embeddings

RiWalk [66]. RiWalk designs structural node indicators approximating graph kernels. In a given subgraph $\mathcal{G}_i^k$, the indicator approximating shortest path kernel for node $v_j \in \mathcal{N}_i^k$ is defined as the concatenation of the degrees of v_i and v_j and the shortest path length between them:

$$\phi_{SP}^i(v_j) = \text{b}(d_i) \circ \text{b}(d_j) \circ s_{ij}, v_j \in \mathcal{N}_i^k \tag{27}$$

where $\text{b}(x) = \lfloor \log(x) + 1 \rfloor$ is a logarithmically binning function. The indicator approximating Weisfeiler-Lehman sub-tree kernel is defined as:

$$\phi_{WL}^i(v_j) = \text{b}(\mathbf{l}^{<i,i>}) \circ \text{b}(\mathbf{l}^{<i,j>}) \circ s_{ij}, v_j \in \mathcal{N}_i^k \tag{28}$$

where $\mathbf{l}^{<i,j>}$ is a vector of length $k+1$ whose l-th element is the count of v_j's neighbers at distance l to v_i in $\mathcal{G}_i^k$, i.e.:

$$\mathbf{l}_l^{<i,j>} = |\{v_{j'} \in \mathcal{N}_j | s_{ij'} = l\}|, l = 0, ..., k \tag{29}$$

Then the random walks starting from v_i are performed on each $\mathcal{G}_i^k$. The nodes are relabeled indicated by Eq.(27) or Eq.(28) while only v_i is not relabeled. And embeddings are learned via SGNS on the generated sequences.

NODE2BITS [67]. NODE2BITS is designed for entity resolution on temporal networks. Here we use τ_{ij} to denote the the timestamp of edge e_{ij}. To integrate temporal information, NODE2BITS utilizes temporal random walks in which edges are sampled with non-decreasing timestamps. The following stepping probability is defined to capture short-term transitions in temporal walks:

$$p_s(v_i, v_j) = \frac{\exp(-\tau_{ij}/T)}{\sum_{(v_i, v_{j'})\in\mathcal{E}} \exp(-\tau_{ij'}/T)} \tag{30}$$

where T is the maximal duration between all timestamps. The stepping probability in long-term policy is defined similarly with positive signs. Multiple walks are generated for each edge and the temporal context of different hops Δt for a node can be extracted from the walks. Then structual features and attributes are fused in temporal walks. For each node v_i with a specific Δt, histograms are applied on multi-dimensional features (and node types if the network is heterogeneous) to aggregate information in the neighborhood and they are concatenated as a vector $(\mathbf{H}_{HIST})_i^{\Delta t}$. SimHash [80] is applied by projecting the histogram $(\mathbf{H}_{HIST})_i^{\Delta t}$ to several random hyperplanes for generating binary hashcode $(H_{HASH})_i^{\Delta t}$. The final embeddings are obtained via concatenation on $(H_{HASH})_i^{\Delta t}$ across different Δts.

Remark. The above three methods are very different on their motivations of utilizing structural feature-based random walks. Role2vec assigns roles firstly and then employs random walks with role indicators. It essentially captures proximity between assigned roles. RiWalk relabels the walks in subgraphs to approximate graph kernels. NODE2BITS uses random walks as neighbor feature aggregators.

3.3 Deep Learning Models

Recently, a few works focus on leveraging deep learning techniques to role-oriented network representation learning. Though deep learning can provide more varied and powerful mapping mechanisms, it needs to be trained with more carefully designed structural information guidance.

3.3.1 *Structural Information Reconstruction/Guidance*

DRNE [50]. DRNE is proposed to capture regular equivalence in networks, so it learns node embeddings in a recursive way with the following loss function:

$$\mathcal{L}_{equiv} = \sum_{v_i \in \mathcal{V}} \left\| \mathbf{H}_i - \breve{\mathbf{H}}_i \right\|_2^2 \tag{31}$$

where $\breve{\mathbf{H}}_i$ is the aggregation of the neighbors' embeddings via a layer normalized Long Short-Term Memory. To make the neighbor information available for LNLSTM, for each node v_i, it downsamples a fixed number of neighbors with large degrees and orders them based on the degrees. Denoting their embeddings as $\{\mathbf{H}_{(1)}, ..., \mathbf{H}_{(T)}\}$ the aggregating process is $\breve{\mathbf{H}}_{(t)} = \text{LNLSTM}(\mathbf{H}_{(t)}, \breve{\mathbf{H}}_{(t-1)})$ and finally $\breve{\mathbf{H}}_i = \breve{\mathbf{H}}_{(T)}$.

Additionally, DRNE proposes a degree-guided regularizer to avoid the trivial solution where all embeddings are $\mathbf{0}$. The regularizer is as follows:

$$\mathcal{L}_{deg} = \sum_{v_i \in \mathcal{V}} (\log(d_i + 1) - \text{MLP}_{deg}(\breve{\mathbf{H}}_i))^2 \tag{32}$$

The regularizer with a parameter γ_{deg} is weighed and the whole model is trained via the combined loss:

$$\mathcal{L} = \mathcal{L}_{equiv} + \gamma_{deg}\mathcal{L}_{deg} \tag{33}$$

GAS [68]. Graph Neural Networks have the power to capture structure as they are closely related to Weisfeiler-Lehman (WL) test in some ways [81]. GAS applies a L-layer graph convolutional encoder, in which each layer is :

$$\mathbf{H}^{(l)} = \sigma(\tilde{\mathbf{A}}\mathbf{H}^{(l-1)}\Theta^{(l-1)}) \tag{34}$$

where $\tilde{\mathbf{A}} = \mathbf{A}+\mathbf{I}$ and $\Theta^{(l-1)}$ is the parameter matrix in the l-th layer. The input $\mathbf{H}^{(0)}$ could be $\tilde{\mathbf{A}}$ or an embedding lookup table. Here the sum-pooling propagation rule is applied instead of the original GCN [82] to better distinguish local structures. In fact, more powerful GNNs such as Graph Isomorphic Network [81] may further improve the performance. The key idea for GAS is that using a few critical structural features as the guidance information to train the model. The features are extracted in a similar way proposed in ReFeX but aggregated only once, normalized and not binned. With a MLP model as the decoder to approximate the features, i.e., $\hat{\mathbf{F}} = \text{MLP}_{dec}(\mathbf{H})$. The loss function is:

$$\mathcal{L} = \left\| \mathbf{F} - \hat{\mathbf{F}} \right\|_F^2 \tag{35}$$

RESD [69]. RESD also adopts ReFeX [72] to extract appropriate features $\mathbf{F}_{ReFeX}$. It uses a Variational Auto-Encoder [83] architecture to learn the low-noise and robust representations:

$$\begin{aligned}
\mathbf{Z}_i &= \text{MLP}_{enc}(\mathbf{F}_i) \\
\boldsymbol{\mu}_i &= \mathbf{W}_{\boldsymbol{\mu}}\mathbf{Z}_i + \mathbf{b}_{\boldsymbol{\mu}} \\
\log(\boldsymbol{\sigma}_i) &= \mathbf{W}_{\boldsymbol{\sigma}}\mathbf{Z}_i + \mathbf{b}_{\boldsymbol{\sigma}} \\
\mathbf{H}_i &= \boldsymbol{\mu}_i + \boldsymbol{\sigma}_i \odot \boldsymbol{\epsilon}, \boldsymbol{\epsilon} \sim \text{Gaussian}(\mathbf{0}, \mathbf{I}) \\
\hat{\mathbf{F}}_i &= \text{MLP}_{dec}(\mathbf{H}_i)
\end{aligned} \tag{36}$$

The VAE model is trained via feature reconstruction. A degree-guided regularizer Eq.(32) designed in DRNE [50] is introduced in RESD for preserving topological characteristics. The combined objective is as follows:

$$\mathcal{L} = \left\| \mathbf{F} - \hat{\mathbf{F}} \right\|_F^2 + \gamma_{deg}\mathcal{L}_{deg} \tag{37}$$

GraLSP [70]. GraLSP is a GNN framework integrating local structural patterns that can be employed on role-oriented tasks. For a node v_i, it captures structural patterns by generating w random walks starting from v_i with length l_w: $\mathcal{W}_i = \{\omega_{i1}, ..., \omega_{iw}\}$, and then anonymizes them $\text{aw}(\omega)$ [84]. Each anonymous walk $\text{aw}(\omega)$ is represented as an embedding lookup table $\mathbf{u}_{\text{aw}(\omega)}$. Then the aggregation of neighborhood representation is designed as follows:

$$\begin{aligned}
(\mathbf{H}_{nei})_i^{(l)} &= \text{MEAN}_{\omega \in \mathcal{W}_i, j \in [1, \lfloor \frac{2l_w}{|\omega|} \rfloor]}(\alpha_{i,\omega}^{(l)}(\mathbf{a}_{i,\omega}^{(l)} \odot \mathbf{H}_{\omega_j}^{(l-1)})) \\
\mathbf{H}_i^{(l)} &= \text{ReLU}(\mathbf{W}_{self}^{(l)}\mathbf{H}_i^{(l-1)} + \mathbf{W}_{nei}^{(l)}(\mathbf{H}_{nei})_i^{(l)})
\end{aligned} \tag{38}$$

where $\mathbf{W}_{self}$ and $\mathbf{W}_{nei}$ are trainable parameter matrices. $\alpha_{i,\omega}^{(l)}$ is learned attention values based on their local structure:

$$\alpha_{i,\omega}^{(l)} = \frac{\exp(\text{SLP}_{att}(\mathbf{u}_{\text{aw}(\omega)}))}{\sum_{\omega' \in \mathcal{W}_i} \exp(\text{SLP}_{att}(\mathbf{u}_{\text{aw}(\omega')}))} \tag{39}$$

$\text{SLP}(\cdot)$ denotes a single-layer perceptron. $\mathbf{a}_{i,\omega}^{(l)}$ is the amplification coefficients:

$$\mathbf{a}_{i,\omega}^{(l)} = \text{SLP}_{amp}(\mathbf{u}_{\text{aw}(\omega)}) \tag{40}$$

To preserve proximities between nodes, the loss function in DeepWalk [77] is leveraged:

$$\begin{aligned}
\mathcal{L}_{prox} = -\sum_{v_i \in \mathcal{V}} \sum_{v_j \in \mathcal{N}_i} &(\log\sigma(\mathbf{H}_i\mathbf{H}_i^\top) \\
&-\gamma_{neg}\mathbb{E}_{v_k \sim P_n(v)}\left[\log\sigma(\mathbf{H}_i\mathbf{H}_k^\top)\right])
\end{aligned} \tag{41}$$

After L aggregations, the embeddings are $\mathbf{H} = \mathbf{H}^{(L)}$. To capture structural similarities between nodes, GraLSP designs the following loss:

$$\begin{aligned} \mathcal{L}_{struc} = &- \sum_{v_i \in \mathcal{V}, \omega_j, \omega_k, \omega_s \in \mathcal{W}_i} \log \sigma(\mathbf{u}_j^\top \mathbf{u}_k - \mathbf{u}_k^\top \mathbf{u}_s), \\ s.t.\ & \hat{p}(\omega_j|v_i) > \hat{p}(\omega_j|\mathcal{G}), \hat{p}(\omega_k|v_i) > \hat{p}(\omega_k|\mathcal{G}), \\ & \hat{p}(\omega_n|v_i) < \hat{p}(\omega_n|\mathcal{G}). \end{aligned} \tag{42}$$

where $\hat{p}(\cdot)$ is the empirical distribution of anonymous walks:

$$\begin{aligned} \hat{p}(\omega_j|v_i) &= \frac{\sum_{\omega \in \mathcal{W}_i} \mathbb{I}(\mathrm{aw}(\omega) = \omega_j)}{w} \\ \hat{p}(\omega_j|\mathcal{G}) &= \frac{\sum_{i=1}^{n} \hat{p}(\omega_j|v_i)}{n} \end{aligned} \tag{43}$$

The objective is to combine the two losses with a trade-off parameter γ_{struc}:

$$\mathcal{L} = \mathcal{L}_{prox} + \gamma_{struc}\mathcal{L}_{struc} \tag{44}$$

GCC [71]. GCC is a pre-train Graph Neural Network model, which represents a node as $\mathbf{H}_i$ by encoding its node-centric subgraph and tries to distinguish the similar subgraphs from the dissimilar ones. It aims to leverage several large-scale networks to train the ability of a GNN-based model to discriminate node substructures in an unsupervised manner. When the scale of the pre-trained datasets is large enough, GCC has the power to recognize local connective patterns of nodes, and the representations generated by it can measure structural similarities. While other GNN methods (such as [85], [86]) either concentrates on community and proximity, or trains model in a supervised manner. So we introduce GCC as role-oriented network embedding method. Specifically, for each node v_i in a network, GCC extracts its k-hop reachable neighborhood as $\mathcal{G}_i^k$, and leverages the Graph Isomorphic Network [81] as the encoder whose convolutional layer is:

$$\mathbf{H}^{(l)} = \mathrm{MLP}_{GIN}((\mathbf{A} + (1+\epsilon)\cdot \mathbf{I})\mathbf{H}^{(l-1)}) \tag{45}$$

where ϵ could be a learnable or fixed parameter and the input attributes of GIN is initialized as the eigenvectors of $\mathcal{G}_i^k$. A similar subgraph instance of v_i is induced based on the nodes in the random work with restart starting from v_i. K dissimilar instances are subgraphs induced in the same way but starting from other nodes which could be in other networks. The representations $\{\mathbf{x}_0, ..., \mathbf{x}_K\}$ of these $K+1$ instances are generated via another GIN encoder. The representation of the similar instance is denoted as $\mathbf{x}^+ \in \{\mathbf{x}_0, ..., \mathbf{x}_K\}$, and GCC is pre-trained by a contrastive learning method called InfoNCE [87]:

$$\mathcal{L} = \sum_{v_i \in \mathcal{V}} -\log \frac{\exp(\mathbf{H}_i \mathbf{x}^+/\iota)}{\sum_{j=0}^{K} \exp(\mathbf{H}_i \mathbf{x}_j/\iota)} \tag{46}$$

where ι is a hyper-parameter.

Remark. These deep methods incorporate model traditional role-related concepts used in shallow methods with deep learning techniques to map structural information into non-linear latent representations. DRNE learns regular equivalence and reconstructs node degrees. GAS uses structural features to guide the training of graph convolutional networks. RESD combines variants of DRNE and ReFeX via a VAE architecture. GraLSP reconstructs the similarities based on anonymous walks. And GCC is a pre-train model which encodes node-centric subgraphs and is pre-trained via a constrastive learning manner.

4 Experimental Evaluation

In this section, we show the comprehensive analysis of these popular role-oriented embedding methods on widely used benchmarks. The experimental evaluations are conducted from the perspectives of both efficiency and effectiveness. To analyze the efficiencies of these methods, we compare their running time for generating node representations on both real-world and synthetic networks with varying sizes. To evaluate the effectiveness of these methods, we select four tasks for the evaluation including (1) the classification experiment based on the ground-truth labels of datasets by comparing the Micro-F1 and Macro-F1 scores, (2) the clustering experiment by comparing some clustering indices with K-Means model in an unsupervised manner. (3) the visualization experiment by plotting the node representations in a 2-D space to observe the relationships between node embeddings and their roles. (4) the top-k similarity search experiment to see if nodes in the same role are mapped into close position in the embedding space.

It's worth noting that we do not select link prediction for the experimental study, mainly because (1) only limited previous role-oriented NE methods evaluate the performance of this task (see Table 2). Thus, it is difficult to make a consistent and fair comparison for all these methods. (2) It has been demonstrated that global information, i.e., roles, is less useful than local information, i.e., proximity, in link prediction task [88]. So we believe that this task is more suitable for proximity-preserving rather than role-oriented node embeddings.

4.1 Datasets

We perform our experiments on the following datasets which form unweighted and undirected networks to illustrate the potential of role-oriented network embedding methods in capturing roles. Based on the type of conducted experiments, these networks are divided in two groups. One group of networks, which include **Brazil**, **Europe**, **USA**, **Reality-call**, **Actor** and **Film**, are for node classification and clustering in which the nodes are labeled based on some role-related rules. The other networks are all Wiki-talk networks in which some nodes have the same but very rare role. These Wiki-talk networks are for top-k similarity search. We show some statistical characteristics from various aspects of these networks in Table 3 and Table 4, respectively. More details about the datasets can be found in the Supplementary Materials.

4.2 Experimental Settings

Although different methods with different embedding mechanism have been proposed, according to our proposed two-level classification taxonomy discussed in Section **??**, for each class of role-oriented network embedding, we choose 4 methods to analyze the performance on different role related tasks. In specific, RolX [38], RIDεRs [58], GraphWave [59]

TABLE 3
Statistic of the networks for node classification and clustering.

Dataset	# Nodes	# Edges	# Classes	Density(%)	Mean Degree	Average CC	Transitivity
Brazil	131	1, 074	4	12.6130	16.3969	0.6364	0.4497
Europe	399	5, 995	4	7.5503	30.0501	0.5393	0.3337
USA	1, 190	13, 599	4	1.9222	22.8555	0.5011	0.4263
Reality-call	6, 809	7, 697	3	0.0332	2.2608	0.0178	0.0024
Actor	7, 779	26, 733	4	0.0888	6.8917	0.0790	0.0156
Film	27, 312	122, 706	4	0.0329	8.9855	0.1180	0.0278

TABLE 4
Statistic of the networks for top-k similarity search.

Dataset	# Nodes	# Edges	# Bots	# Admins	Density(%)	Mean Degree	Average CC	Transitivity
ht-wiki-talk	446	758	24	0	0.7638	3.3991	0.0941	0.0085
br-wiki-talk	1, 049	2, 330	35	8	0.4239	4.4423	0.1998	0.0410
cy-wiki-talk	2, 101	3, 610	31	16	0.1636	3.4365	0.1579	0.0090
oc-wiki-talk	3, 064	4, 098	43	4	0.0873	2.6749	0.0994	0.0023
eo-wiki-talk	7, 288	14, 266	120	21	0.0537	3.9149	0.1206	0.0085
gl-wiki-talk	7, 935	19, 887	12	14	0.6318	5.0125	0.4913	0.0037

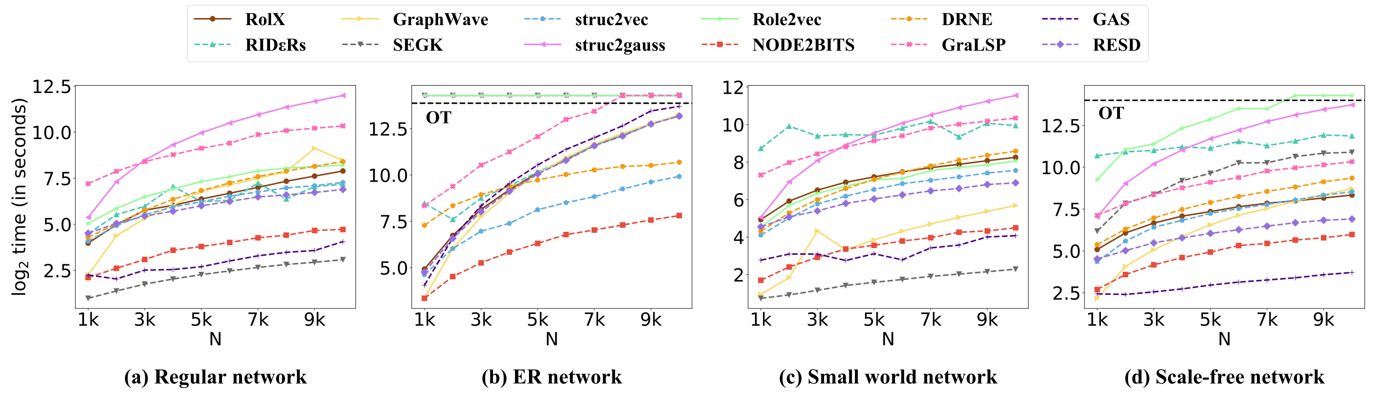


Fig. 5. Running time of 12 methods on four different types of synthetic networks with different sizes.

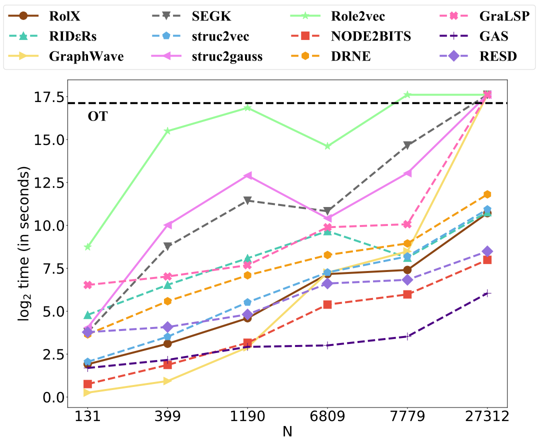


Fig. 6. Running time of 12 methods on the six real networks.

and SEGK [56] belong to the low-rank matrix factorization. struc2vec [39], struc2gauss [64], Role2Vec [65] and NODE2BITS [67] are all based on random walk. DRNE [50], GraLSP [70], GAS [68] and RESD [69] pertain to the scope of deep learning. In the subsequent experiments, all the parameters are fine-tuned. Note that for Role2vec, we use motif-count features as the default setting, but it causes the out-of-memory issue in large networks and we use node degrees to circumvent. We release the datasets and source code used in the experiments on Github .

4.3 Efficiency analysis

All experiments are performed on a machine with Intel(R) Xeon(R) CPU E5-2680 v4 at 2.40GHz and 125GB RAM. In this experiment, by ignoring the differences in some implementation details, we report the running time of these methods on the above six real networks. The results are shown in Fig 6. The y-axis represents the average logarithmic time (seconds) of 10 times running of each method on one specific network and x-axis is the size (number of nodes) of the network. If the value is above the dotted line, it means that the cost of corresponding method is beyond our tolerance. Generally speaking, the methods based on deep learning are relatively efficient compared to others, especially for the methods that need higher-order features. GAS [68], RESD [69] and NODE2BITS [67] are the three most efficient methods and RolX [38] also has competitive results. struc2gauss [64], SEGK [56] and Role2vec [65] cost more time to learn the node embeddings since they need to compute higher-order features, e.g., motif. In particular, as

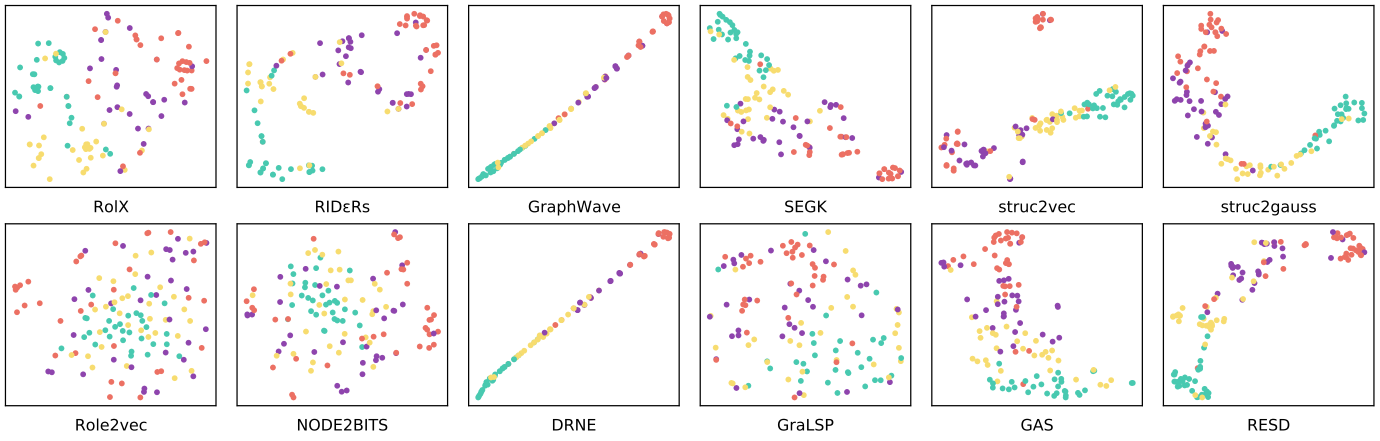


Fig. 7. Visualization results of 12 methods using t-SNE on the Air Brazil network. The points denote the nodes and the same color indicates that they belong to the same label.Two nodes are more closer to each other if their embeddings are role similar.

TABLE 5
Node classification average F1 score on different networks.

Method	Brazil		Europe		USA		Reality-call		Actor		Film	
	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro
RolX	0.749	0.741	0.556	0.546	0.623	0.617	0.593	0.383	0.465	0.451	**0.494**	**0.396**
RIDεRs	**0.790**	**0.783**	0.539	0.519	0.626	0.618	0.635	0.396	**0.470**	0.448	0.479	0.380
GraphWave	**0.762**	**0.753**	0.526	0.491	0.517	0.469	**0.839**	**0.516**	0.251	0.180	OM	OM
SEGK	0.723	0.718	0.536	0.524	0.615	0.607	**0.839**	**0.514**	**0.479**	**0.460**	OM	OM
struc2vec	**0.766**	**0.757**	**0.577**	**0.572**	0.599	0.594	0.593	0.376	0.463	**0.456**	0.474	0.365
struc2gauss	0.730	0.715	**0.585**⋆	**0.580**⋆	**0.641**	**0.632**	0.603	0.391	0.456	0.449	OT	OT
Role2vec	0.385	0.358	0.362	0.347	0.467	0.455	0.541	0.354	0.277†	0.277†	0.278†	0.243†
NODE2BITS	0.593	0.577	0.490	0.477	0.583	0.573	0.524	0.354	0.442	0.424	**0.542**⋆	**0.428**⋆
DRNE	0.716	0.700	0.542	0.521	0.601	0.588	0.630	0.478	0.462	0.449	0.453	0.339
GAS	0.757	0.750	**0.583**	**0.574**	**0.668**⋆	**0.659**⋆	**0.841**⋆	**0.529**⋆	**0.480**⋆	**0.466**⋆	**0.490**	**0.380**
RESD	**0.797**⋆	**0.791**⋆	0.557	0.544	**0.640**	**0.634**	0.607	0.411	**0.476**	**0.463**	**0.516**	**0.415**
GraLSP	0.510	0.490	0.455	0.422	0.535	0.523	0.454	0.300	0.341	0.316	OM	OM

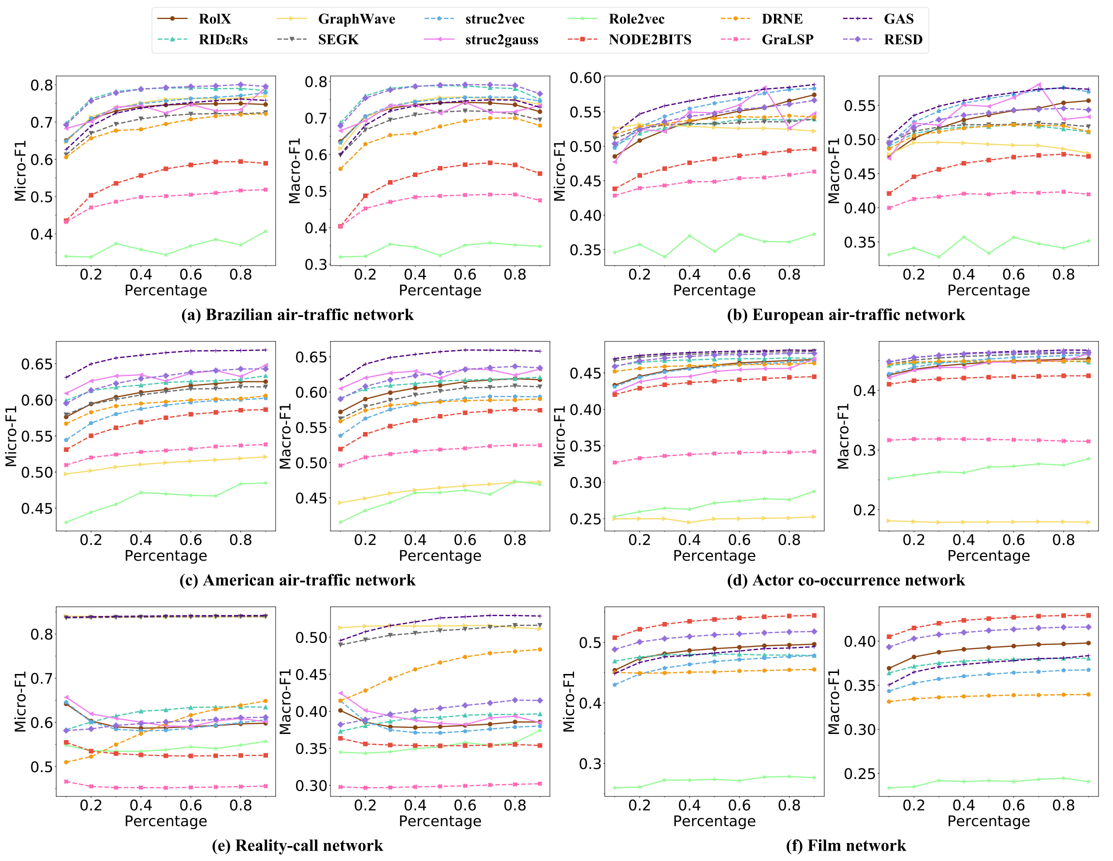
the most classical method proposed in the early stage, RolX still shows competitive in efficiency.

However, the numbers of nodes in these networks are quite different in a random way. In order to show the efficiency of different methods more fairly, we generate four different types of networks with linear variation of the size. The details are as following. Regular network. It is a regular graph in which each node has the same number of neighbors and we set it as 3. ER network [89]. It is based the ER random graph and we set the probability of having a link between any two nodes as 0.1. Small world network. It comes from the popular Watts–Strogatz model [90] and for each link, we set it rewiring probability as 0.5. Scale-free network. It is based on the Barabási–Albert model [91] with preferential attachment, for each new nodes, the number of its links is set as 3.

For each type of network, we vary the number of nodes N from $1,000$ to $10,000$. The running time of the baselines is shown in Fig 5. From these results, some conclusions can be drawn. Firstly, the ER network is relatively dense for its fixed link probability. The average degree of nodes in larger networks is larger correspondingly. In this type of network, NODE2BITS has significant efficiency advantage than all others, and it also achieves competitive results in other three types of networks. Secondly, SEGK is very efficient on the regular and small world networks but time-consuming on the ER and scale-free networks. Thirdly, GAS is the most effective on the scale-free networks and has second results only to the SEGK. Besides, RolX and RESD rank in the middle of this experiments for they all are based on ReFex [72] and reconstruct the features with different methods. GAS is one of the most efficient on almost all the networks. GAS reconstructs a small number of primary features which leads to few parameters and epochs for loss convergence. NODE2BIT consumes little time on all the networks stably. Because its feature aggregation process, which is based on random walk and the Simhash mapping process, is almost linear to the number of edges.

4.4 Visualization analysis

Visualization can help to understand the performance of the different methods intuitively. It is also a common way for understanding the network structure and the methods. In this section, we show the t-SNE results of these methods based on the Brazil network in Fig 7. It can be observed that, RIDεRs [58], GraphWave [59], SEGK [56], struc2vec [39], struc2gauss [64], DRNE [50], GAS [68] and RESD [69] can be more accurate to identify the labels compared with others. The cyan nodes correspond to the airports with large passenger flow and all these methods can embed it effectively except NODE2BITS [67]. The red nodes represent the marginal airports. These nodes are usually hard to be well identified for they are more scattered in the network. Therefore, these methods can not identify this kind of nodes



(a) Brazilian air-traffic network (b) European air-traffic network

(c) American air-traffic network (d) Actor co-occurrence network

(e) Reality-call network (f) Film network

Fig. 8. Classification results of the 12 role-oriented network embedding methods on the six real world networks.The performance is based on the Micor-F1 and Macor-F1 with different percentages of training set, which is the average values of 20 runs.

well. Besides, although we expect to embed nodes with same role into close points in latent space, it is also still the demand nodes should also be slightly different even they are in the same role. From this view, struc2gauss, SEGK and GAS are better selections. GAS benefits from the low-dimensional features which leads to the same-role embeddings gathering. The embeddings of SEGK and struc2gauss shape some clumps since they are directly constrained by effective similarities.

4.5 Classification results

Here we report the performance of the classification experiment on the six networks. In detail, with the learned node embeddings of each method, we randomly select 70% of the nodes as the training set and the others as the test set and take the logistic regression classifier to learn the labels of nodes. The results are shown in Table 5. It represents the Micro-F1 and Macro-F1 value of classification results. To avoid losing of generality, we report the average results of 20 times of independent runs on each network.

As shown in Table 5, for each column, we mark the values of methods with significant advantages, i.e. the top results of these methods. OM and OT mean that it cannot be calculated for fixed memory and limited time, and the † denotes that Role2vec uses degree features rather than motif features because it is very time-consuming.

For the three air-traffic networks, RESD [69] performs best on Brazil and RIDεRs [58] follows because the ReFeX [72] features used by them can effectively capture structural similarities in small networks. While on the other two networks, GAS [68] and struc2gauss [64] perform well. For the rest three large networks, struc2guass cannot generate node embeddings within limited time, though it performs well in small networks. GAS outperforms others on Reality-call and Actor, while NODE2BITS [67] gets the highest score on Film.

Among all the datasets, we observe that Role2vec [65] obtains the worst performance. This may because that it leverages the features-based random walk and damages the original network structure. It can be applied to link prediction task, but is not proper for role discovery. Though GraLSP [70] leverages the local structural patterns, it concentrates more on node proximity and fails to distinguish roles of nodes.

In Section 3, we have divided these methods into three types: random walk, matrix factorization, and deep learning. In general, deep learning methods perform well on all the networks. For the three small air-traffic networks, random walks (struc2vec [39] and struc2gauss) perform well, while matrix factorization methods get better results on large networks, even if some of them cannot obtain embeddings because of the out of memory error. Overall, deep learning methods are scale-well and can effectively discover roles.

For matrix factorization methods, RIDεRs is better than RolX [38] except on Film, and they both can be applied

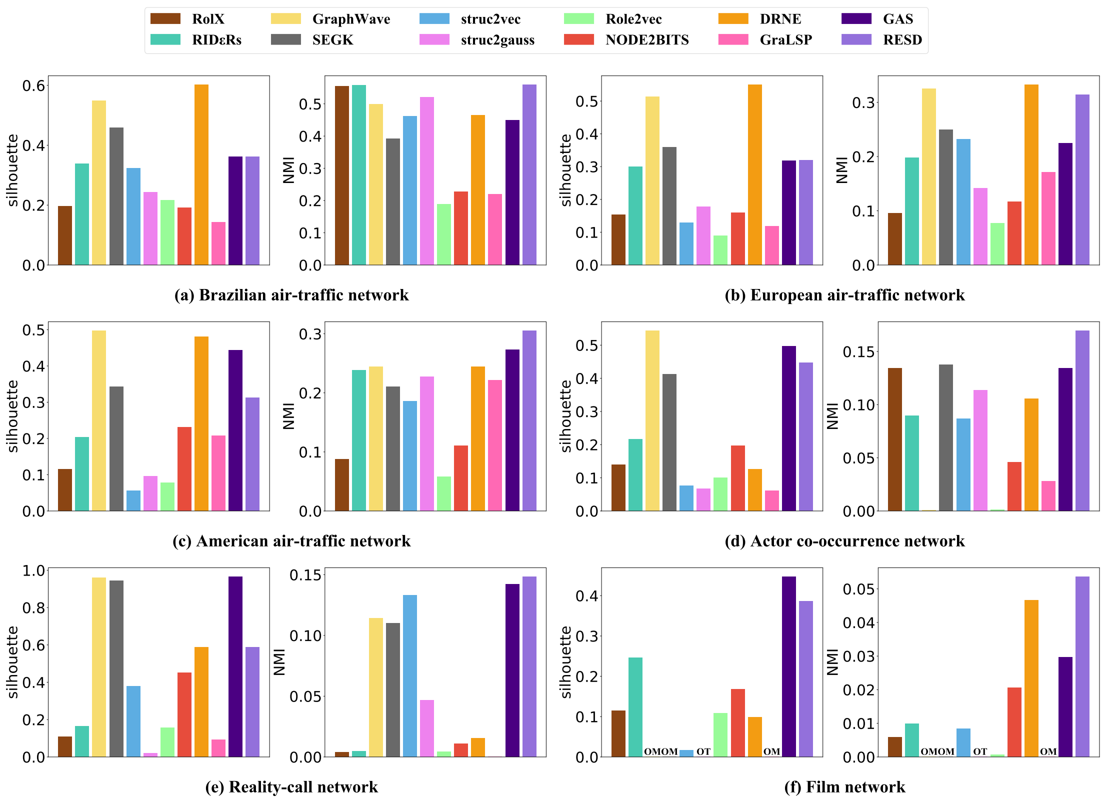


Fig. 9. Clustering results of the 12 role-oriented network embedding methods on the six real world networks. The performance is based on the silhouette coefficient and NMI and evaluated on 10 times run.

to large-scale networks. GraphWave [59] and SEGK [56] perform best on Reality-call but consume too much memory. For random walk methods, struc2vec and struc2gauss show their superiority on small networks, while NODE2BITS is more suitable for large-scale networks. As for deep learning methods, GAS and RESD stay ahead of other approaches on all networks.

As shown in Fig. 6, for RESD, NODE2BITS, and GAS, there is a linear relationship between their computational complexities and network sizes. In Table 5, GAS and RESD perform well on all networks, while NODE2BITS only gets the best score on Film. It is obvious that combining structural features and deep learning models has advantages in both time and classification.

Further more, in order to analyze the influence of different percentages of training set on classification results, we also show the performance of these methods with varying size of training (from 10% to 90%). The results are shown in Fig 8. The experimental settings are same introduced above.

In general, as the ratio of the training set increases, the F1 scores of classification results are improved. On the three air-traffic networks, we observe significant increase, but when we use too many training samples (more than 80%), the scores go down because of over-fitting. The scores on large networks are stable, because small percentage of training samples are sufficient enough to make the linear model constraint.

The scores on Reality-call network are different to others, because the distribution of labels is unbalance, and the Micro-F1 scores are higher than Macro-F1. In general, we observe the scores of Role2vec fluctuate, which means that Role2vec fails for the task of role classification. GraLSP also performs a litter better than Role2vec, but it gets worse on Reality-call. Other methods show similar tendencies, while deep learning methods (GAS and RESD) and NODE2BITS exhibit significant superiority.

4.6 Clustering results

In the social science, the role discovery is usually denoted as an unsupervised task which can be implemented by the unsupervised clustering in machine learning. In this experiment, we study the role clustering problem based on the embedding of different methods. Based on the embedding, we take the k-means algorithm to obtain the clusters, and report the silhouette coefficient [92] and Normalized Mutual Information (NMI) [93] in Fig. 9. The silhouette coefficient is a measure of how similar an object is to its own cluster compared to other clusters. The result only relies on the embeddings themselves rather than true labels. While the NMI measures the similarity between clustering results and real distribution.

We observe that GraphWave [59] obtains high scores on both measurements in air-traffic networks, but in larger networks (Actor and Reality-call), it gets low NMI scores. DRNE [50] shows similar tendency because its aggregation methodology leads to obvious clustering, but may not fit the true distributions. The results of Role2vec [65] and GraLSP [70] are similar to classification, and they fail to cluster roles of nodes. The performances of other random walk and matrix factorization methods are moderate. In general, it can be observed that deep learning methods (GAS [68] and RESD [69]) show overall superiority. They perform well on small networks but significantly outperform others in large-scale networks. This is also consistent with findings of deep learning methods in other domains that deep learning methods can better learn the patterns from larger-scale data.

4.7 Top-k similarity search

We conduct top-k similarity search experiments on the 6 Wiki-talk networks to evaluate the ability to retrieve rare roles of different methods. In specific, after generating role-based embeddings for each node through a candidate method, we find the k most similar users for each bot or administrator by computing the euclidean distance between the embeddings. The mean precision for both bots and administrators on different sizes of the retrieved node list, i.e., different ks, is reported in the Supplementary Materials.

In general, we assume that nodes literally having the same role (such as bots in Wiki-networks) in a network have similar structural patterns. However, there are more bots than administrators in most Wiki-talk networks, almost every method achieves higher precision on searching administrators than that on searching bots when the retrieved list size k is fixed and smaller than or close to the number of administrators. This shows that the structures of bots are more irregular or these methods can not effectively capture the specific structural patterns of bots. The structural representation learning on literal roles is much more difficult than on function-based defined roles (such as the classes in Table 3), since the latent patterns of literal roles are usually not reflected in the plain structural properties.

What's more, the inconsistency between literal roles and node structure is obvious across networks. We can come to this conclusion gradually through the following observations: the proportion of the same literal role in different networks varies greatly as shown in Table 4; none of the compared methods can produce top results on all the networks with k fixed; although the proportion of bots in cy-wiki-talk and oc-wiki-talk networks are very close, the results of almost every method on cy-wiki-talk are better than those on oc-wiki-talk. Thus, the literal roles in different networks are less comparable and transferable to those roles (such as the classes in Table 3) defined directly by node functions.

Though no one method achieves best performance on all the networks, the ranks of each method on a network with different retrieved list size k are usually close. Some methods, such as RIDεRs [58], GraphWave [59] and NODE2BITS [67] show their wider applicability across different networks. NODE2BITS surprisingly achieves good and stable results compared with its performance on node classification and clustering tasks. Its feature aggregation mechanism based on random walks makes it lose some elaborate information but capture more latent structural patterns. Similarly, RIDεRs and GraphWave can also capture more latent structural patterns via ε-equitable refinement and characterizing wavelet distributions, respectively. Role2vec still lacks competitiveness on this task, since it assigns roles based on features firstly and generates embeddings capturing proximities between roles as we argued above. The deep learning methods DRNE [50], GAS [68] and RESD [69] show mediocre performance as they overfit some features due to the deep process of feature reconstruction. While the similarity captured by GraLSP [70] based on anonymous walks is too coarse.

4.8 Discussion on experiments

In this section, we show a variety of experiments on some popular role-oriented network embedding methods. These methods cover all the three major categories and the five minor categories we propose. We evaluate their effectiveness on classification, clustering, visualization and top-k similarity search and efficiency on networks in different scales.

We find that no one method can outperform the others on all tasks. As we argued above, each task has a different focus and the evaluated methods are only suitable for parts of them. On balance, GAS [68] and RESD [69] are good at visualization, node classification and clustering tasks. NODE2BITS [67] show outstanding performance on both top-k similarity search and efficiency test. Role2vec [65] ceases to be effective on these role-based tasks as it assigns roles based on features directly while the embeddings do not capture the structural similarities. It specializes in link prediction for non-direct role-based tasks. Similarity, the structural similarity objective of GraLSP [70] is not powerful enough for role-based tasks. The performance of the other methods fluctuates a lot on different tasks and networks. Thus, it is still an open problem that how we can capture both fine and essential role information effectively and efficiently.

Though some deep learning methods seems good at many experiments, they have not yet addressed the nature of the role. The deep learning manners give them better capabilities on machine learning tasks such as classification and clustering. However, they utilize additional skills to extract structural information, and lack a suitable objective function to embed it without bias and impurities. Thus, deep role-base representation research, which is still in the beginning stage, is promising and needs more attention.

5 APPLICATIONS

Role discovery can complement community detection in network clustering and has always been the focus of social science and network science research. Role oriented network embedding has gradually become one of the focuses of graph machine learning. It can benefit to node ranking, community detection, information spreading and other problems in complex networks. It also contributes to the research and development of graph neural networks in machine learning. So we summarize some important applications of role oriented network embedding as follows.

First, it is beneficial to a variety of network mining tasks. Community detection and role discovery are two complementary tasks from local and global perspective of networks structures respectively [94]. Thus, combining these two tasks can help to achieve better network generation and mutually improve each other. For instance, MMCR has been proposed to integrate community detection and role discovery in a unified model and detect both of them simultaneously for information networks [95]. The proposed method extends the Mixed Membership Stochastic Blockmodel (MMSB) [35] to combine the generative process of both community and role. REACT [51] analyzes the community structure and role discovery under a unified framework and describes their relations via non-negative matrix factorization. Moreover, role-oriented embedding can also help other tasks including link prediction [88], anomaly detection [38], and structural similarity search on network [49].

Second, it can help to analyze the network dynamics and evolution. As we know, nodes with different roles may have inequable influence in information diffusion. RAIN [96] analyzes the effect of users with different roles on their reposting messages and models the generation of diffusion process under a unified probabilistic framework. Then, with a general representation learning on the network, it can effectively improve the performance of cascade [97] and its popularity [98] prediction. Besides, analyzing the roles of nodes could be used to predict the network evolution and its dynamic behaviors [99], [100] and detecting the varying of roles of node could capture the dynamic structural information [101].

Moreover, it has induced some new graph neural networks with more expressive ability. Although GNNs with the message passing are mainly used for supervised node or graph classification and link prediction, some new architectures have been developed to improve their expressive ability with the heterogeneity and roles in the network. Geom-GCN [102] has been proposed to use the role embedding method to obtain the structural neighborhood and a bi-level aggregation in GNN. ID-GNN [85] extracts the ego network centered at one node of the network and takes rounds of heterogeneous message passing. It can compute the shortest path and clustering coefficient of the network. These features are are extremely relevant to the role. GCC [71] also makes use of ego network and then employs pre-training for role-oriented network embedding.

Besides, it can shed light on new patterns discovery in specific networks. Role discovery and identification can be used to predict the social behaviors, identities and the temporal patterns [103]. [104] analyzes the user behaviors in the Blog social network and learn roles from the concepts of activity, influence and competition. It also can predict the sentiment and topics with the proposed definitions of roles. EMBER [62] considers professional role inference in a large-scale email network, so it can capture and distinguish the behavior similarity of nodes automatically. SADE [105] combines role information and subgraph embedding to detect subgraph-level anomalies from financial transaction networks.

6 Future Directions

Although many creative and innovative methods have been proposed for role oriented network embedding, it still faces some problems and challenges to be solved in this field. In this section, we will discuss these challenges as future directions.

Role-oriented embedding on dynamic networks. Most methods reviewed in this survey are for static networks. However, real-world networks are naturally dynamic and continuously streaming over time with evolving structures. Thus role-oriented NE methods for dynamic networks are of fundamentally practical and theoretical importance. There have only been a few approaches for role discovery on dynamic networks e.g., Role-Dynamics [106] and DyNMF [101]. However, some questions remain open. For example, how to separate the dynamic networks into different snapshots and how to model the deletion of nodes and edges. Thus, additional investigation is needed to extend current NE methods for dynamic scenarios.

Other types of embedding spaces. Beside to the Euclidean space, a variety of methods project the nodes into the Hyperbolic space [107] in proximity preserved network representation learning. For role oriented embedding, as we know, the only work is Hyperboloid [108] which extends the struc2vec [39] with the hyperboloid model. Considering the non-Euclidean space is quite suitable for network embedding because of the power-law distribution of network data, it is interesting and meaningful to have some in-depth analysis and understanding in utilizing hyperbolic neural network for this problem.

Construction of larger-scale benchmarks. All the methods for role oriented network embedding are evaluated on relatively small-scale networks data with thousands of nodes (seen in section 4). However, real-world networks are often of a massive scale, e.g., there are billions of users in social networks. Constructing larger-scale benchmark datasets is very important to evaluate existing approaches in effectiveness, efficiency and robustness, and also beneficial for researchers to develop new models.

Interpretation on roles and role-oriented embeddings. In social science, roles often correspond to social identifications, e.g., students and teachers in a school. Real-world networks may not contain such information and thus are difficult to understand the discovered roles. The lack of interpretability of roles and role-oriented embeddings may significantly impact our ability to gain insights into these roles. However, so far, there have not been much work focusing on discovering roles that are interpretable. One exception is RolX [38], and it attempts to explain roles based on several structural measures such as centrality. More research is needed to discover roles and learn embeddings that are more interpretable.

Deep theoretical analysis. Important but not the final, network embedding for role discovery lacks theoretical analysis. Although some methods stem from certain type of equivalence relation, the theoretical analysis has been ignored. Moreover, current methods cannot capture universal node representations and provide the upper/lower bound of the expressive ability. It is expected to have solid theoretical analysis in this filed similar to that in graph neural networks [81].

7 Conclusions

Role-oriented network embedding can complement the category of network representation learning. It has gradually become one of most important research focuses in network embedding. In this survey, we have proposed a general understanding mechanism for role-oriented network embedding approaches and a two-level classification ontology based on different embedding principles. Using this ontology, we categorize a series of popular methods into different groups. Then, we review the principles and innovations of selected representative embedding methods in different categories. We further conduct comprehensive experiments to evaluate these representative methods, including role classification and clustering, top-k similarity search, visualization and their efficiency test. Last, we summarize the important applications of this problem and outline some future directions. We believe that this survey can help to understand and deepen role-oriented network embedding and it will attract more attention from network science and deep learning.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China (61902278).

REFERENCES

[1] S. H. Strogatz, "Exploring complex networks," *Nature*, vol. 410, no. 6825, pp. 268–276, 2001.

[2] J. Gao, B. Barzel, and A.-L. Barabási, "Universal resilience patterns in complex networks," *Nature*, vol. 530, no. 7590, pp. 307–312, 2016.

[3] V. Martínez, F. Berzal, and J.-C. Cubero, "A survey of link prediction in complex networks," *ACM Computing Surveys (CSUR)*, vol. 49, no. 4, pp. 1–33, 2016.

[4] Y. Liu, T. Safavi, A. Dighe, and D. Koutra, "Graph summarization methods and applications: A survey," *ACM Computing Surveys (CSUR)*, vol. 51, no. 3, pp. 1–34, 2018.

[5] Z.-K. Zhang, C. Liu, X.-X. Zhan, X. Lu, C.-X. Zhang, and Y.-C. Zhang, "Dynamics of information diffusion and its applications on complex networks," *Physics Reports*, vol. 651, pp. 1–34, 2016.

[6] M. Á. Serrano and M. Boguna, "Clustering in complex networks. i. general formalism," *Physical Review E*, vol. 74, no. 5, p. 056114, 2006.

[7] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.

[8] M. Rosvall and C. T. Bergstrom, "An information-theoretic framework for resolving community structure in complex networks," *Proceedings of the National Academy of Sciences*, vol. 104, no. 18, pp. 7327–7331, 2007.

[9] J. Cheng, M. Chen, M. Zhou, S. Gao, C. Liu, and C. Liu, "Overlapping community change-point detection in an evolving network," *IEEE Transactions on Big Data*, vol. 6, no. 1, pp. 189–200, 2020.

[10] R. A. Rossi and N. K. Ahmed, "Role discovery in networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 4, pp. 1112–1131, 2014.

[11] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3-5, pp. 75–174, 2010.

[12] R. Merton, R. Merton, and M. P. Company, *Social Theory and Social Structure*, ser. American studies collection. Free Press, 1968.

[13] G. Liu, Y. Wang, and M. Orgun, "Social context-aware trust network discovery in complex contextual social networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 26, no. 1, 2012.

[14] S. Gilpin, T. Eliassi-Rad, and I. Davidson, "Guided learning for role discovery (glrd) framework, algorithms, and applications," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2013, pp. 113–121.

[15] M. Chen, K. Kuzmin, and B. K. Szymanski, "Community detection via maximization of modularity and its variants," *IEEE Transactions on Computational Social Systems*, vol. 1, no. 1, pp. 46–65, 2014.

[16] X. Li, M. K. Ng, and Y. Ye, "Multicomm: Finding community structure in multi-dimensional networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 4, pp. 929–941, 2013.

[17] N. Stanley, S. Shai, D. Taylor, and P. J. Mucha, "Clustering network layers with the strata multilayer stochastic block model," *IEEE Transactions on Network Science and Engineering*, vol. 3, no. 2, pp. 95–105, 2016.

[18] P. Chopade and J. Zhan, "A framework for community detection in large networks using game-theoretic modeling," *IEEE Transactions on Big Data*, vol. 3, no. 3, pp. 276–288, 2017.

[19] L. Yang, X. Cao, D. Jin, X. Wang, and D. Meng, "A unified semi-supervised community detection framework using latent space graph regularization," *IEEE Transactions on Cybernetics*, vol. 45, no. 11, pp. 2585–2598, 2014.

[20] Y. Pei, N. Chakraborty, and K. Sycara, "Nonnegative matrix tri-factorization with graph regularization for community detection in social networks," in *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[21] X. Ma and D. Dong, "Evolutionary nonnegative matrix factorization algorithms for community detection in dynamic networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 5, pp. 1045–1058, 2017.

[22] X. Ma, D. Dong, and Q. Wang, "Community detection in multi-layer networks using joint nonnegative matrix factorization," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 2, pp. 273–286, 2018.

[23] C. Tu, X. Zeng, H. Wang, Z. Zhang, Z. Liu, M. Sun, B. Zhang, and L. Lin, "A unified framework for community detection and network representation learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 6, pp. 1051–1065, 2018.

[24] F. Liu, S. Xue, J. Wu, C. Zhou, W. Hu, C. Paris, S. Nepal, J. Yang, and P. S. Yu, "Deep learning for community detection: Progress, challenges and opportunities," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, 2020, pp. 4981–4987.

[25] H. Feng, J. Tian, H. J. Wang, and M. Li, "Personalized recommendations based on time-weighted overlapping community detection," *Information and Management*, vol. 52, no. 7, pp. 789–800, 2015.

[26] J. Zheng, S. Wang, D. Li, and B. Zhang, "Personalized recommendation based on hierarchical interest overlapping community," *Information Sciences*, vol. 479, pp. 55–75, 2019.

[27] Z. Lu, X. Sun, Y. Wen, G. Cao, and T. La Porta, "Algorithms and applications for community detection in weighted networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 11, pp. 2916–2926, 2014.

[28] S. Fortunato and D. Hric, "Community detection in networks: A user guide," *Physics Reports*, vol. 659, pp. 1–44, 2016.

[29] D. Jin, Z. Yu, P. Jiao, S. Pan, P. S. Yu, and W. Zhang, "A survey of community detection approaches: From statistical modeling to deep learning," *arXiv preprint arXiv:2101.01669*, 2021.

[30] F. Lorrain and H. C. White, "Structural equivalence of individuals in social networks," *The Journal of Mathematical Sociology*, vol. 1, no. 1, pp. 49–80, 1971.

[31] D. R. White and K. P. Reitz, "Graph and semigroup homomorphisms on networks of relations," *Social Networks*, vol. 5, no. 2, pp. 193–234, 1983.

[32] P. W. Holland and S. Leinhardt, "An exponential family of probability distributions for directed graphs," *Journal of the American Statistical Association*, vol. 76, no. 373, pp. 33–50, 1981.

[33] K. Nowicki and T. A. B. Snijders, "Estimation and prediction for stochastic blockstructures," *Journal of the American Statistical Association*, vol. 96, no. 455, pp. 1077–1087, 2001.

[34] K. Faust and S. Wasserman, "Blockmodels: Interpretation and evaluation," *Social Networks*, vol. 14, no. 1-2, pp. 5–61, 1992.

[35] E. M. Airoldi, D. Blei, S. Fienberg, and E. Xing, "Mixed membership stochastic blockmodels," in *Advances in Neural Information Processing Systems*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds., vol. 21, 2009.

[36] A. Arockiasamy, A. Gionis, and N. Tatti, "A combinatorial approach to role discovery," in *2016 IEEE 16th International Conference on Data Mining (ICDM)*, 2016, pp. 787–792.

[37] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, p. P10008, 2008.

[38] K. Henderson, B. Gallagher, T. Eliassi-Rad, H. Tong, S. Basu, L. Akoglu, D. Koutra, C. Faloutsos, and L. Li, "Rolx: structural role extraction & mining in large graphs," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2012, pp. 1231–1239.

[39] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo, "struc2vec: Learning node representations from structural identity," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 385–394.

[40] J. Qiu, Y. Dong, H. Ma, J. Li, C. Wang, K. Wang, and J. Tang, "Netsmf: Large-scale network embedding as sparse matrix factorization," in *Proceedings of the World Wide Web*, 2019, pp. 1509–1520.

[41] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 5, pp. 833–852, 2018.

[42] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Network representation learning: A survey," *IEEE Transactions on Big Data*, vol. 6, no. 1, pp. 3–28, 2020.

[43] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2020.

[44] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2021.

[45] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *Knowledge-Based Systems*, vol. 151, pp. 78–94, 2018.

[46] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 9, pp. 1616–1637, 2018.

[47] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *IEEE Data Engineering Bulletin*, vol. 40, no. 3, pp. 52–74, 2017.

[48] H. Chen, B. Perozzi, R. Al-Rfou, and S. Skiena, "A tutorial on network embeddings," *arXiv preprint arXiv:1808.02590*, 2018.

[49] R. A. Rossi, D. Jin, S. Kim, N. K. Ahmed, D. Koutra, and J. B. Lee, "On proximity and structural role-based embeddings in networks: Misconceptions, techniques, and applications," *ACM Transactions on Knowledge Discovery from Data*, vol. 5, no. 14, pp. 1–37, 2020.

[50] K. Tu, P. Cui, X. Wang, P. S. Yu, and W. Zhu, "Deep recursive network embedding with regular equivalence," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2018, pp. 2357–2366.

[51] Y. Pei, G. Fletcher, and M. Pechenizkiy, "Joint role and community detection in networks via l 2, 1 norm regularized nonnegative matrix tri-factorization," in *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 2019, pp. 168–175.

[52] J. Jin, M. Heimann, D. Jin, and D. Koutra, "Towards understanding and evaluating structural node embeddings," *arXiv preprint arXiv:2101.05730*, 2021.

[53] P. W. Holland and S. Leinhardt, "An exponential family of probability distributions for directed graphs," *Journal of the American Statistical Association*, vol. 76, no. 373, pp. 33–50, 1981.

[54] D. R. White and K. P. Reitz, "Graph and semigroup homomorphisms on networks of relations," *Social Networks*, vol. 5, no. 2, pp. 193–234, 1983.

[55] J. Guo, L. Xu, and J. Liu, "Spine: Structural identity preserved inductive network embedding," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, 2019, pp. 2399–2405.

[56] G. Nikolentzos and M. Vazirgiannis, "Learning structural node representations using graph kernels," *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[57] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[58] P. V. Gupte, B. Ravindran, and S. Parthasarathy, "Role discovery in graphs using global features: algorithms, applications and a novel evaluation strategy," in *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, 2017, pp. 771–782.

[59] C. Donnat, M. Zitnik, D. Hallac, and J. Leskovec, "Learning structural node embeddings via diffusion wavelets," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2018, pp. 1320–1329.

[60] R. A. Rossi, N. K. Ahmed, E. Koh, S. Kim, A. Rao, and Y. Abbasi-Yadkori, "A structural graph representation learning framework," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020, pp. 483–491.

[61] M. Heimann, H. Shen, T. Safavi, and D. Koutra, "Regal: Representation learning-based graph alignment," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 117–126.

[62] D. Jin, M. Heimann, T. Safavi, M. Wang, W. Lee, L. Snider, and D. Koutra, "Smart roles: inferring professional roles in email networks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019, pp. 2923–2933.

[63] B. Shi, C. Zhou, H. Qiu, X. Xu, and J. Liu, "Unifying structural proximity and equivalence for network embedding," *IEEE Access*, vol. 7, pp. 106 124–106 138, 2019.

[64] Y. Pei, X. Du, J. Zhang, G. Fletcher, and M. Pechenizkiy, "struc2gauss: Structural role preserving network embedding via gaussian embedding," *Data Mining and Knowledge Discovery*, vol. 34, no. 4, pp. 1072–1103, 2020.

[65] N. K. Ahmed, R. A. Rossi, J. B. Lee, T. L. Willke, R. Zhou, X. Kong, and H. Eldardiry, "Role-based graph embeddings," pp. 1–1, 2020.

[66] M. Xuewei, G. Qin, Z. Qiu, M. Zheng, and Z. Wang, "Riwalk: Fast structural node embedding via role identification," in *2019 IEEE International Conference on Data Mining (ICDM)*, 2019, pp. 478–487.

[67] D. Jin, M. Heimann, R. A. Rossi, and D. Koutra, "Node2bits: Compact time-and attribute-aware node representations for user stitching," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2019, pp. 483–506.

[68] X. Guo, W. Zhang, W. Wang, Y. Yu, Y. Wang, and P. Jiao, "Role-oriented graph auto-encoder guided by structural information," in *International Conference on Database Systems for Advanced Applications*, 2020, pp. 466–481.

[69] W. Zhang, X. Guo, W. Wang, Q. Tian, L. Pan, and P. Jiao, "Role-based network embedding via structural features reconstruction with degree-regularized constraint," *Knowledge-Based Systems*, p. 106872, 2021.

[70] Y. Jin, G. Song, and C. Shi, "Gralsp: Graph neural networks with local structural patterns," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 4361–4368.

[71] J. Qiu, Q. Chen, Y. Dong, J. Zhang, H. Yang, M. Ding, K. Wang, and J. Tang, "Gcc: Graph contrastive coding for graph neural network pre-training," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2020, pp. 1150–1160.

[72] K. Henderson, B. Gallagher, L. Li, L. Akoglu, T. Eliassi-Rad, H. Tong, and C. Faloutsos, "It's who you know: graph mining using recursive structural features," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2011, pp. 663–671.

[73] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, no. 5, pp. 465–471, 1978.

[74] P. Drineas and M. W. Mahoney, "On the nyström method for approximating a gram matrix for improved kernel-based learning," *Journal of Machine Learning Research*, vol. 6, no. 12, pp. 2153–2175, 2005.

[75] C. K. Williams and M. Seeger, "Using the nyström method to speed up kernel machines," in *Advances in Neural Information Processing Systems*, 2001, pp. 682–688.

[76] R. Jin, V. E. Lee, and H. Hong, "Axiomatic ranking of network role similarity," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2011, pp. 922–930.

[77] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014, pp. 701–710.

[78] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 855–864.

[79] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their

compositionality," in *Advances in Neural Information Processing Systems*, 2013, pp. 3111–3119.

[80] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing*, 2002, pp. 380–388.

[81] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *International Conference on Learning Representations*, 2019.

[82] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations*, 2017.

[83] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *International Conference on Learning Representations*, 2014.

[84] S. Ivanov and E. Burnaev, "Anonymous walk embeddings," in *International Conference on Machine Learning*, 2018, pp. 2186–2195.

[85] J. You, J. Gomes-Selman, R. Ying, and J. Leskovec, "Identity-aware Graph Neural Networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

[86] J. You, R. Ying, and J. Leskovec, "Position-aware graph neural networks," in *International Conference on Machine Learning*, 2019, pp. 7134–7143.

[87] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[88] T. Lyu, Y. Zhang, and Y. Zhang, "Enhancing the network embedding quality with structural similarity," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017, pp. 147–156.

[89] P. Erdös and A. Rényi, "On random graphs i," *Publicationes Mathematicae Debrecen*, vol. 6, p. 290, 1959.

[90] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world'networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[91] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

[92] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987.

[93] L. Danon, A. Diaz-Guilera, J. Duch, and A. Arenas, "Comparing community structure identification," *Journal of Statistical Mechanics: Theory and Experiment*, 2005.

[94] Y. Pei, *On local and global graph structure mining*, ser. SIKS Dissertation Series. Technische Universiteit Eindhoven, 2020, no. 2020-05.

[95] T. Chen, L.-A. Tang, Y. Sun, Z. Chen, H. Chen, and G. Jiang, "Integrating community and role detection in information networks," in *Proceedings of the 2016 SIAM International Conference on Data Mining*, 2016, pp. 72–80.

[96] Y. Yang, J. Tang, C. W.-k. Leung, Y. Sun, Q. Chen, J. Li, and Q. Yang, "RAIN: Social Role-Aware Information Diffusion," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, no. 1, 2015, pp. 367–373.

[97] J. Caverlee, X. B. Hu, M. Lalmas, W. Wang, A. Sankar, X. Zhang, A. Krishnan, and J. Han, "InF-VAE: A variational autoencoder framework to integrate homophily and influence in diffusion prediction," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020, pp. 510–518.

[98] F. Zhou, X. Xu, K. Zhang, G. Trajcevski, and T. Zhong, "Variational Information Diffusion for Probabilistic Cascades Prediction," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 2020, pp. 1618–1627.

[99] S. Jiang and H. Chen, "Natergm: A model for examining the role of nodal attributes in dynamic social media networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 3, pp. 729–740, 2016.

[100] "Exploring the transition behavior of nodes in temporal networks based on dynamic community detection," *Future Generation Computer Systems*, vol. 107, pp. 458–468, 2020.

[101] Y. Pei, J. Zhang, G. Fletcher, and M. Pechenizkiy, "DyNMF: Role Analytics in Dynamic Social Networks," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, 2018, pp. 3818–3824.

[102] H. Pei, B. Wei, K. C.-C. Chang, Y. Lei, and B. Yang, "Geom-GCN: Geometric Graph Convolutional Networks," in *International Conference on Learning Representations*, 2020.

[103] S. Zhang, X. Liang, and J. Qi, "A review on role identification methods in social networks," *Chinese Journal of Computers*, vol. 40, no. 3, pp. 649–673, 2017.

[104] A. Zygmunt, J. Koźlak, B. Gliwa, M. Stojkow, D. Żuchowska–Skiba, and Y. Demazeau, "Achieving and maintaining important roles in social media," *Information Processing and Management*, vol. 57, no. 3, p. 102223, 2020.

[105] Y. Pei, F. Lyu, W. v. Ipenburg, and M. Pechenizkiy, "Subgraph anomaly detection in financial transaction networks," in *ACM International Conference on AI in Finance*, 2020.

[106] R. Rossi, B. Gallagher, J. Neville, and K. Henderson, "Role-dynamics: Fast mining of large dynamic networks," in *Proceedings of the 21st International Conference on World Wide Web*, 2012, p. 997–1006.

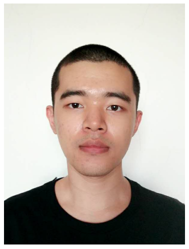
[107] W. Peng, T. Varanka, A. Mostafa, H. Shi, and G. Zhao, "Hyperbolic Deep Neural Networks: A Survey," *arXiv preprint arXiv:2101.04562*, 2021.

[108] L. Wang, Y. Lu, C. Huang, and S. Vosoughi, "Embedding node structural role identity into hyperbolic space," in *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*, 2020, p. 2253–2256.

Pengfei Jiao received the Ph.D. degree in computer science from Tianjin University, Tianjin, China, in 2018. He is a lecture with the Center of Biosafety Research and Strategy of Tianjin University. His current research interests include complex network analysis, data mining and graph neural network and applications of statistical network model. He has published more than 50 international journals and conference papers.

Xuan Guo is pursuing a doctoral degree at the College of Intelligence and Computing, Tianjin University, P. R. China. His current research interests include complex network analysis, role discovery, network representation learning and percolation model.

Ting Pan received the Bachelor degree from Xiamen University in 2020. She is currently pursuing a master's degree at the School of Computer Science and Technology, Tianjin University. Her current research interests include complex network analysis and role-based network representation learning.



Wang Zhang received the Bachelor degree from Tianjin University in 2018. He is currently pursuing a master's degree at the School of Computer Science and Technology, Tianjin University. His current research interests include complex network analysis and network embedding.

Yulong Pei received the Ph.D. degree in computer science from Eindhoven University of Technology, Eindhoven, the Netherlands, in 2020. He is a postdoc researcher in the Department of Mathematics and Computer Science, Eindhoven University of Technology. His current research interests include graph mining, social network analysis, and anomaly detection.

Supplementary Materials for A Survey on Role-Oriented Network Embedding

Pengfei Jiao, Xuan Guo, Ting Pan, Wang Zhang, and Yulong Pei

Appendix A Details about Datasets

Air-traffic networks [1]. They are from the aviation network of Brazil, USA and Europe, respectively. In each network, nodes represent airports and edges represent existing aircraft routes between airports. As the labeled networks, the nodes are labeled according to the activity level of airports, which forms 4 classes in each network. In the following paper, we will simply denote the 3 networks as Brazil, USA and Europe.

Reality-call network [2]. It records $52,050$ phone calls of $6,809$ users from September 2004 to January 2005. After processing the dataset, we get a static network with $6,809$ nodes and $7,697$ edges, in which nodes represent users and edges represent communication between users. The nodes are divided into different categories according to the call frequency of users.

Actor co-occurrence network [3] (shortly as Actor). This is a subgraph of the Film network that only contains actors. The nodes are sorted in accordance with the number of words on their Wikipedia pages, and then are divided into 4 labels which can be regarded as a measure of their influence.

English-language film network [3] (shortly as Film). The network assigns a label to each node to represent the role attributes of the node. In other words, it indicates whether the node is a film, director, actor or writer. The edge denotes that the two nodes have appeared on the same Wikipedia page.

Wiki-talk networks [4]. These networks are extracted from talk pages of Wikipedia in different languages. The nodes denote Wikipedia users and two nodes have an edge if their corresponding users have communications. Each network has a very small number of nodes which are bots and administrators in reality. Top-k similarity search on these bots and administrators requires role-oriented network embedding methods to have ability to capture more significant or intrinsic structural patterns. Here we use 6 of the 28 networks because their scale is suitable for both employment and comparison of most baseline methods. For each used network, we denote it as ”wiki-talk” shortly with the ISO 639 code of the language as the prefix (ht-Haitian, br-Breton , cy-Welsh, oc-Occitan, eo-Esperanto, gl-Galician).

For all the above networks, we report the network density, mean degree, average clustering coefficient (CC) and transitivity of these networks for simply depicting their degree of structural sparsity and heterogeneity in the main manuscript. These characteristics are usually used to analyze roles in networks [5]. Network density is the ratio of the actual number of edges in a network to the upper limit of the edges number that can be accommodated, which is used to describe the density of the connected edges between nodes in the network. Degree centrality is the most direct measure of node centrality in network analysis. The assumption behind this indicator is that important nodes have more connections. The clustering coefficient is a measure of the degree to which nodes in a network tend to cluster together. Transitivity is also called global clustering coefficient. It is the proportion of triangle structure in the network to all possible triangle structures.

Appendix B Results of Top-k Similarity Search

The experimental results of top-k similarity search is shown in Table 1. We highlight the top 3 results on each network in bold while the best values are marked with an asterisk. We note Role2vec [6] with ”(M)” when it leverages motif-count features but ”(D)” when it leverages node degrees as features for avoiding out-of-memory problem.

References

[1] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo, “struc2vec: Learning node representations from structural identity,” in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 385–394.

[2] X. Guo, W. Zhang, W. Wang, Y. Yu, Y. Wang, and P. Jiao, “Role-oriented graph auto-encoder guided by structural information,” in *International Conference on Database Systems for Advanced Applications*, 2020, pp. 466–481.

[3] M. Xuewei, G. Qin, Z. Qiu, M. Zheng, and Z. Wang, “Riwalk: Fast structural node embedding via role identification,” in *2019 IEEE International Conference on Data Mining (ICDM)*, 2019, pp. 478–487.

[4] J. Sun, J. Kunegis, and S. Staab, “Predicting user roles in social networks using transfer learning with feature transformation,” in *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, 2016, pp. 128–135.

[5] A. Bartal and G. Ravid, “Member behavior in dynamic online communities: role affiliation frequency model,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 9, pp. 1773–1784, 2019.

[6] N. K. Ahmed, R. A. Rossi, J. B. Lee, T. L. Willke, R. Zhou, X. Kong, and H. Eldardiry, “Role-based graph embeddings,” pp. 1–1, 2020.

TABLE 1
Top-k similarity search (mean precision) on Wiki-talk Networks.

Network	Method	k=5		k=10		k=25		k=50		k=100	
		bot	admin	bot	admin	bot	admin	bot	admin	bot	admin
ht-wiki-talk	RolX	31.67	−−	22.92	−−	14.00	−−	**13.00**	−−	**11.46**	−−
	RID$\mathcal{E}$Rs	**36.67**	−−	**26.67**	−−	14.33	−−	**13.33⋆**	−−	**14.54⋆**	−−
	GraphWave	**40.00**	−−	26.25	−−	**16.67**	−−	11.42	−−	9.42	−−
	SEGK	34.17	−−	**26.67**	−−	13.17	−−	10.58	−−	7.79	−−
	struc2vec	26.67	−−	18.75	−−	11.83	−−	10.83	−−	8.25	−−
	struc2gauss	30.83	−−	21.67	−−	11.67	−−	9.67	−−	7.96	−−
	Role2vec (M)	10.83	−−	6.25	−−	4.33	−−	3.33	−−	2.79	−−
	NODE2BITS	**42.50⋆**	−−	**28.33⋆**	−−	**17.00⋆**	−−	11.67	−−	8.33	−−
	DRNE	25.00	−−	20.83	−−	13.50	−−	10.42	−−	7.67	−−
	GAS	28.33	−−	**26.67**	−−	14.67	−−	12.08	−−	**10.79**	−−
	RESD	27.50	−−	17.50	−−	11.83	−−	**13.25**	−−	10.12	−−
	GraLSP	23.33	−−	21.25	−−	**14.83**	−−	11.50	−−	8.50	−−
br-wiki-talk	RolX	21.14	**47.50⋆**	17.14	**42.50⋆**	**14.97**	**27.00⋆**	**14.06⋆**	**14.00**	**8.77**	**7.00**
	RID$\mathcal{E}$Rs	16.57	**40.00**	16.57	**31.25**	12.80	**26.50**	11.43	**14.00**	8.49	**7.00**
	GraphWave	21.71	35.00	17.43	26.25	12.34	19.50	10.34	10.00	6.31	5.00
	SEGK	18.29	27.50	16.57	25.00	13.71	23.50	12.06	12.75	7.43	**7.00**
	struc2vec	21.14	35.00	17.14	**30.00**	**14.86**	**26.00**	11.60	**14.00**	8.31	**7.00**
	struc2gauss	**23.43**	17.50	**22.00⋆**	20.00	**15.89⋆**	16.50	11.89	13.75	8.71	**7.00**
	Role2vec (M)	7.43	7.50	6.86	10.00	6.63	10.00	4.86	7.50	3.74	5.37
	NODE2BITS	**25.71⋆**	27.50	**20.29**	21.25	12.91	18.50	10.11	10.50	7.86	6.00
	DRNE	13.71	17.50	13.43	13.75	9.37	10.50	7.89	7.00	7.46	3.50
	GAS	**24.57**	10.00	**18.57**	17.50	13.94	13.00	**13.14**	10.50	**8.89**	6.12
	RESD	15.43	**42.50**	16.29	26.25	13.60	25.00	**14.00**	**14.00**	**9.03⋆**	**7.00**
	GraLSP	5.14	5.00	5.71	5.00	5.37	3.50	5.43	3.50	5.97	3.25
cy-wiki-talk	RolX	8.39	38.75	10.32	34.37	9.55	26.25	7.55	19.75	6.55	11.87
	RID$\mathcal{E}$Rs	14.19	45.00	**18.06⋆**	38.12	**13.03**	**34.50**	8.13	**21.62**	7.48	**13.12⋆**
	GraphWave	**15.48⋆**	**57.50**	**16.13**	**44.37**	11.10	32.75	8.13	17.87	6.29	10.12
	SEGK	**14.84**	**67.50⋆**	**15.81**	**58.75⋆**	11.10	**34.75**	**10.26**	**21.50**	**7.58**	**12.56**
	struc2vec	12.26	52.50	13.55	**49.37**	**13.81⋆**	**36.50⋆**	**12.26⋆**	20.37	**7.74⋆**	11.06
	struc2gauss	12.26	28.75	10.65	22.50	10.84	21.25	8.00	**22.25⋆**	6.03	**13.12⋆**
	Role2vec (D)	3.87	6.25	1.94	4.37	1.94	2.75	1.68	1.87	1.42	1.50
	NODE2BITS	**14.84**	**55.00**	**15.81**	43.75	**11.87**	29.25	**10.32**	19.87	**7.55**	11.81
	DRNE	5.16	25.00	4.19	18.75	2.84	11.50	2.58	6.62	3.13	3.56
	GAS	14.19	30.00	10.97	25.62	10.58	19.25	9.42	11.50	6.48	7.25
	RESD	9.68	37.50	9.03	30.62	8.90	30.25	7.29	18.12	6.32	10.87
	GraLSP	5.16	6.25	4.52	5.00	2.71	6.00	1.94	4.87	1.84	3.62
oc-wiki-talk	RolX	2.79	0.00	4.88	5.00	4.65	2.00	3.44	1.00	2.44	0.50
	RID$\mathcal{E}$Rs	7.44	**10.00⋆**	6.51	5.00	5.02	2.00	3.72	1.00	2.72	0.75
	GraphWave	**8.37**	**10.00⋆**	**7.91**	**12.50**	**6.70**	**7.00**	**5.81**	3.50	3.63	1.75
	SEGK	**9.77⋆**	**10.00⋆**	**7.91**	**15.00⋆**	6.60	**10.00⋆**	4.79	**6.00⋆**	2.77	**3.00⋆**
	struc2vec	5.12	**10.00⋆**	6.51	**15.00⋆**	6.05	**10.00⋆**	**5.95⋆**	**6.00⋆**	3.79	**3.00⋆**
	struc2gauss	**9.30**	5.00	**8.14⋆**	2.50	**8.09⋆**	2.00	5.77	4.00	**4.58⋆**	**3.00⋆**
	Role2vec (D)	1.40	0.00	0.93	0.00	1.49	0.00	1.86	0.50	1.51	0.50
	NODE2BITS	6.51	**10.00⋆**	6.05	7.50	**6.88**	6.00	**5.81**	**4.50**	**4.51**	2.50
	DRNE	0.93	5.00	1.16	2.50	1.21	3.00	1.30	1.50	1.40	0.75
	GAS	**8.37**	0.00	7.67	2.50	6.23	2.00	5.67	1.00	**4.14**	0.50
	RESD	2.79	5.00	3.72	5.00	3.72	2.00	3.53	1.00	2.42	0.50
	GraLSP	4.65	0.00	3.02	0.00	2.70	0.00	2.09	0.00	1.93	0.25
eo-wiki-talk	RolX	10.33	**42.86⋆**	9.58	**39.05**	8.53	**35.62**	7.00	**29.52⋆**	6.51	**16.86**
	RID$\mathcal{E}$Rs	7.83	28.57	7.75	29.52	6.20	25.33	4.60	26.67	6.29	**18.10⋆**
	GraphWave	12.50	34.29	**11.08**	**35.71**	8.93	**35.05**	6.80	24.10	5.98	13.38
	SEGK	OM	OM	OM	OM	OM	OM	OM	OM	OM	OM
	struc2vec	**11.50**	39.05	10.25	34.76	**9.87⋆**	34.67	**8.30⋆**	**28.48**	**7.46⋆**	15.71
	struc2gauss	8.17	**42.86⋆**	9.00	**40.95⋆**	8.40	**35.81⋆**	5.48	26.10	3.63	**17.48**
	Role2vec (D)	2.17	3.81	2.33	2.38	1.73	0.95	1.62	0.48	1.67	0.43
	NODE2BITS	10.33	35.24	8.75	34.29	7.73	28.76	6.40	22.38	5.52	13.57
	DRNE	6.00	36.19	5.83	27.14	5.40	19.81	4.77	12.00	4.17	7.05
	GAS	**11.00**	31.43	**11.67**	28.57	**9.57**	17.71	**8.05**	14.67	**6.59**	10.90
	RESD	**14.50⋆**	**42.86⋆**	**12.33⋆**	35.24	**9.70**	34.67	7.12	**28.57**	6.02	18.1
	GraLSP	8.67	11.43	7.33	9.52	7.43	7.43	**7.37**	6.10	**7.42**	4.62
gl-wiki-talk	RolX	0.00	17.14	0.83	22.14	0.33	27.14	**1.83**	23.29	0.92	12.86
	RID$\mathcal{E}$Rs	0.00	18.57	1.67	10.00	1.67	**36.00⋆**	1.67	**26.00⋆**	**2.17**	**13.00⋆**
	GraphWave	**6.67**	**41.43**	3.33	**37.14**	**2.00**	**30.29**	1.67	19.43	0.83	9.93
	SEGK	OM	OM	OM	OM	OM	OM	OM	OM	OM	OM
	struc2vec	**8.33⋆**	**47.14⋆**	**5.00**	**51.43⋆**	**2.00**	**35.14⋆**	1.17	22.00	0.67	11.14
	struc2gauss	3.33	18.57	2.50	21.43	1.67	20.86	1.67	**23.71**	1.17	**13.00⋆**
	Role2vec (D)	0.00	0.00	0.00	1.43	0.00	0.57	0.33	0.43	0.17	0.43
	NODE2BITS	**8.33⋆**	**44.29**	**5.83**	**37.86**	**4.33**	27.43	**2.50**	22.00	**2.17**	12.29
	DRNE	1.67	30.00	0.83	26.43	1.00	14.29	0.83	7.86	0.83	4.29
	GAS	3.33	24.29	1.67	23.57	1.33	15.14	1.50	9.86	0.83	5.57
	RESD	0.00	21.43	0.83	23.57	1.00	23.71	1.33	**25.43**	1.08	**13.00⋆**
	GraLSP	6.33	2.86	**6.08⋆**	3.33	**5.83⋆**	3.62	**5.82**	3.24	**6.42⋆**	2.76