Computer Science Review 38 (2020) 100296

Contents lists available at ScienceDirect

Computer Science Review

journal homepage: www.elsevier.com/locate/cosrev

Review article

Network embedding: Taxonomies, frameworks and applications

Mingliang Hou [a], Jing Ren [a], Da Zhang [b], Xiangjie Kong [c], Dongyu Zhang [a], Feng Xia [d,*]

[a] Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, School of Software, Dalian University of Technology, Dalian 116620, China
[b] Department of Electrical and Computer Engineering, University of Miami, 5452 Coral Gables, Miami, FL, 33124, USA
[c] College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, China
[d] School of Engineering, IT and Physical Sciences, Federation University Australia, Ballarat, VIC 3353, Australia

ARTICLE INFO

Article history:
Received 14 June 2019
Received in revised form 11 July 2020
Accepted 5 August 2020
Available online 26 August 2020

Keywords:
Network science
Network embedding
Heterogeneity
Dynamics

ABSTRACT

Networks are a general language for describing complex systems of interacting entities. In the real world, a network always contains massive nodes, edges and additional complex information which leads to high complexity in computing and analyzing tasks. Network embedding aims at transforming one network into a low dimensional vector space which benefits the downstream network analysis tasks. In this survey, we provide a systematic overview of network embedding techniques in addressing challenges appearing in networks. We first introduce concepts and challenges in network embedding. Afterwards, we categorize network embedding methods using three categories, including static homogeneous network embedding methods, static heterogeneous network embedding methods and dynamic network embedding methods. Next, we summarize the datasets and evaluation tasks commonly used in network embedding. Finally, we discuss several future directions in this field.

Contents

* Corresponding author.
E-mail address: f.xia@ieee.org (F. Xia).

https://doi.org/10.1016/j.cosrev.2020.100296
1574-0137/

1. Introduction

Networks are a general data structure that reflects the symmetric and asymmetrical relationships among discrete objects in the real world. Many real-world networks such as social networks [1], language networks [2], wireless networks [3], collaboration networks [4–6] and biological networks [7] contain large amount of information in a specific domain. Analyzing these networks brings challenges to primary machine learning algorithms and frameworks. Typical network analysis tasks include node classification [8], link prediction [9], node clustering [10,11] and network visualization [12].

Mathematically, a network can be defined as $N = (V, E)$. Here, V is the vertex set representing entities and edge set E reflects relationships of entities. A common way to represent network is adopting the adjacency matrix to substitute linking relationships of nodes. A majority of complex characteristics, however, have emerged with the consistent development of networks. These characteristics, such as sparsity, heterogeneity and dynamics also bring challenges to the existing network analysis framework [13]. Therefore, adjacency matrix will result in high computational complexity and low scalability when extracting knowledge from large-scale complex networks.

Recently, network embedding techniques have attracted increasing attention. The goal of network embedding is transforming input network into a low-dimensional space while preserving as much information as possible. The essence of this process is to extract proper features from the input network for the subsequent downstream network analysis tasks. Different network embedding strategies devote to amassing diversified sets of features. After embedding, most network analysis tasks can be solved effectively with conventional machine learning methods. Fig. 1 shows an example of embedding a network into a 2D vector space in the DeepWalk [1].

The development of the network embedding techniques is accompanied by the ever-increasing complexity of networks. Previously, network embedding methods mainly aimed at reducing high dimensionality of networks by utilizing matrix factorization techniques. These networks are constructed manually by linking the non-graph structured entities. Conventional methods construct network by connecting the entities in the vertex set according to their pairwise feature similarities and then embed the network into a low-dimensional space while preserving the neighboring structural information at the same time. Among these methods, Isomap [14], locally linear embedding (LLE) [15] and Laplacian Eigenmaps [16] transform the real-world non-relational data into a network by computing the similarity of entities (K-nearest neighbor). The weight of a link can be obtained by applying the heat kernel function [17]. Then, a specific matrix-factorization based method is utilized to project the original matrix into a low-dimensional output. Reducing the dimensionality of data and preserving their geometric structural information are the essence of these methods.

With the advance of data science [18] and network science, information generated by networks becomes rich and heterogeneous. The preceding network embedding methods preserve well-defined structural information [19], but cannot meet the requirements of recent network analysis tasks. The reason is that previous networks are constructed manually. Relationships among nodes are defined clearly in the original feature space. The constructing of networks can be regarded as a pre-processing step. Therefore, some novel network embedding methods such as DeepWalk [1], LINE [20], TADW [21] and LANE [22] are proposed to preserve diverse information such as structural information (e.g., 1st-order, 2nd-order, etc.), node information (e.g., text information, attributive information, etc.) and network properties (e.g., structural balance property in signed networks, etc.). These innovative methods transform the network into a low-dimensional space while preserving rich semantic and structural information compared with previous methods.

1.1. Challenges

Sparsity, scalability, heterogeneity and dynamics of existing networks give rise to major obstacles in network embedding studies.

1.1.1. Sparsity

Compared with previously defined non-relational data manifold representation networks, current networks are naturally-formed [19]. Many nodes and links may not be observed due to limitations in collection technology and privacy protection. As a result, sparsity caused by the missing information is not rare in current networks.

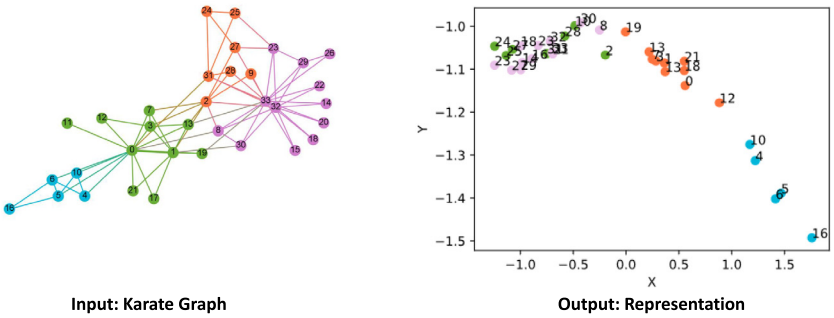

Fig. 1. An example of DeepWalk embedding on a karate network.
Source: The figure is adapted from [1].

1.1.2. Scalability

Large-scale networks bring another challenge to current network researches. Billions of nodes and their attributive information lead to high computational cost and thus require abundant hardware support. Therefore, it is an effective way to disseminate computational tasks on different servers. However, due to the inherent dependence of the network, it is very complicated to design a scalable algorithm while ensuring the consistency of the network structure.

1.1.3. Heterogeneity

As mentioned above, a node of network may contain various types of information (e.g., textual information, label information, etc.). The different types of nodes and links will pose great challenges in designing embedding algorithm. For example, sampling strategy is different between homogeneous network and heterogeneous networks. Mapping different types of nodes and links into low-dimensional vectors while preserving heterogeneity is of great importance for network embedding.

1.1.4. Dynamics

As is known, dynamics is an unavoidable topic in network science. The nodes and links representing the real-world entities and relationships evolve constantly. A traditional technique of processing dynamic networks is to continuously perform the off-line method until the network typologies alter. This strategy, however, is impractical in processing large-scale data. Novel methods or algorithms which can capture the dynamic features and the evolving properties of network effectively will be highly desirable.

In this survey, we propose a novel taxonomy to classify the current network embedding methods from the perspective of network evolvement. Considering the complexity and evolution of the network, we divide the network embedding methods into a three-layer classification framework in our survey. According to this categorization, we systematically analyze the mainstream network embedding algorithms including a wide range of networks: from static to dynamic, from homogeneous to heterogeneous, from topology information to additional information.

1.2. Major differences with other related surveys

In this survey, we aim to review the recent network embedding methods according to network evolvement. Yan et al. [23] introduce graph embedding techniques which are similar to network embedding. However, 'graph' discussed in this survey refers to the manually constructed graph with well-defined relationship information among nodes. In this survey, we pay more attention to naturally-formed networks such as social networks [24], biological networks and citation networks [25]. Naturally-formed networks possess more implicit proximity between nodes, sufficient to pose significant challenges for embedding networks into low-dimensional vector space. Cai et al. [26] and Goyal et al. [27] provide comprehensive survey of graph embedding based on the techniques. We agree with the viewpoint that the technique adopted in one embedding work depends on the network it handles. For instance, random walk-based techniques cannot perform well on heterogeneous networks [28,29]. Therefore, in this survey, we organize network embedding works according to characteristics of network rather than techniques. Cui et al. [19] organize network embedding works from the perspective of network inference. However, network reconstruction is an important objective in the embedding process. In this survey, we consider both of network reconstruction and network inference application tasks of network embedding (see Datasets and evaluations). Wang et al. [30] devote to reviewing the knowledge graph embedding which is far more different from the network embedding. The knowledge graph embedding method such as TransE [31], pay more attention to embedding the triplet $\langle h, r, t\rangle$ into a low-dimensional vector. In essence, knowledge graph itself is different from network we described in this survey.

1.3. Organization

The rest of this paper is organized as follows. In Section 2, we provide notations and definitions mentioned in the following discussions. Section 3 presents a hierarchical classification of network embedding methods and explains the reason for this classification. Sections 4 and 5 analyze the network embedding algorithms according to the categorization. In Section 6, we first introduce the commonly used datasets and evaluation tasks in network embedding. Then, we implement six methods on node classification, node clustering, link prediction and visualization tasks by using real-world dataset. In Section 7, we propose six future directions in network embedding. Finally, we summarize the whole survey in Section 8.

2. Notations and essential definitions

In order to better understand the algorithms and techniques discussed in the following sections, we first introduce several commonly used notations and essential definitions in network embedding, as shown in Table 1 in this section.

Network. A network is defined as $\mathcal{N}^{(t)} = (V^{(t)}, E^{(t)}, V_T^{(t)}, E_T^{(t)})$, where $V^{(t)}$ and $E^{(t)}$ represent vertex set and edge set in the network at t time-stamp respectively. $V_T^{(t)}$ and $E_T^{(t)}$ denote number of vertex types and edge types at t time-stamp respectively.

Homogeneous network. A homogeneous network is defined as $\mathcal{N}_{homo}^{(t)} = (V^{(t)}, E^{(t)}, V_T^{(t)}, E_T^{(t)})$, where $V_T^{(t)} = E_T^{(t)} = 1$. All vertices and edges in $\mathcal{N}$ belong to a single type.

Table 1
A summary of notations.

Notations	Descriptions
$\mathcal{N}^{(t)}$	A network at time-stamp of t
$V^{(t)}$	The vertex set of a network at time-stamp of t
$E^{(t)}$	The edge set of a network at time-stamp of t
$p(k)$	k-order proximity
$V_T^{(t)}$	The number of vertex types in a network at time-stamp of t.
$E_T^{(t)}$	The number of edge types in a network at time-stamp of t.
A	Adjacency matrix of network.
D	Degree matrix of network.
$\mathbf{e_i}$	Embedding result of vertex v_i

Heterogeneous network. A heterogeneous network is defined as $\mathcal{N}_{hete}^{(t)} = (V^{(t)}, E^{(t)}, V_T^{(t)}, E_T^{(t)})$, where $V_T^{(t)} > 1$ or/and $E_T^{(t)} > 1$.

Static network. A static network is defined as $\mathcal{N}_{static}^{(t)} = (V^{(t)}, E^{(t)}, V_T^{(t)}, E_T^{(t)})$, where $\mathcal{N}_{static}^{(t)} = \mathcal{N}_{static}^{(t+1)}$.

Dynamic network. A dynamic network is defined as $\mathcal{N}_{dyn}^{(t)} = (V^{(t)}, E^{(t)}, V_T^{(t)}, E_T^{(t)})$, where $\mathcal{N}_{dyn}^{(t)} \neq \mathcal{N}_{dyn}^{(t+1)}$. Any changes in the element of $\mathcal{N}$ including V, E, V_T, E_T can be regarded as the evolvement of the network.

Adjacency matrix. For a network $\mathcal{N}$ with the vertex set V and $|V| = n$, the adjacency matrix of $\mathcal{N}$ is a square $n \times n$ matrix A. The value of element A_{ij} equals to 1 indicating that there is an edge from vertex i to vertex j.

Degree matrix. For a network $\mathcal{N}$ with the vertex set V and $|V| = n$, the degree matrix D for $\mathcal{N}$ is a $n \times n$ diagonal matrix defined as:

$$d_{i,j} := \begin{cases} deg(v_i) & if \quad i = j \\ 0 & otherwise \end{cases} \tag{1}$$

where the degree $deg(v_i)$ of v_i counts the number of edges terminating at v_i.

1st-order proximity. For each vertex pair (v_i, v_j), the **1st-order** proximity represents the relationship strength between the nodes [20]. The 1st-order proximity can be measured by the weight of the edge between v_i and v_j.

2nd-order proximity. The **2nd-order** proximity reflects the similarity of two vertices' 1st-order adjacent nodes [20]. In other words, the 2nd-order proximity counts the number of common neighbors between two different vertices.

K-order proximity. The **K-order** proximity refers to the $k(k \geq 3)$-step relations between v_i and v_j which can be measured by the transition probability from v_i to v_j [32].

In fact, the 1st-order proximity, the 2nd-order proximity and the high-order proximity can be measured by a unified format: k-step transition probability which can be represented by the k-order adjacency matrix of a network. In general, if A is the adjacency matrix of network, D is the degree matrix of network, the k-order proximity can be defined as:

$$p(k) = [A^{-1}D]^k \tag{2}$$

Network embedding (NE). For a network $\mathcal{N}^{(t)} = (V^{(t)}, E^{(t)}, V_T^{(t)}, E_T^{(t)})$, the task of network embedding is to learn the mapping that can project each node: $v \mapsto \mathbf{e}_v \in \mathbb{R}^d$. Here, $\mathbf{e}_v$ is the output embedding vector and $d \ll |V|$. A transformation that can project input networks into low-dimensional spaces is the target of network embedding methods. After this process, the structural and attributive properties of the original network are not only preserved but also can be used to reconstruct the actual network.

3. Categorization of network embedding techniques

In this section, we propose a novel hierarchical category to classify the existing network embedding techniques from the perspective of network evolvement. As mentioned above, networks are primarily used to represent the small-scale non-relational data which can reflect the high-dimensional properties of the data previously. However, with the advance of Internet and Web technologies, an increasing number of naturally-formed networks have emerged such as social networks [33], language networks and collaboration networks [34–36]. Network embedding methods target to preserve information (e.g., high order proximity information) among nodes which are not explicitly defined in these networks.

Originally, the network embedding techniques started from static, homogeneous networks aiming at capturing basic topology information: the neighboring information between nodes (1st-order proximity). Then, researchers found that direct information between nodes cannot describe the global structural features when the scale of the network becomes larger and larger. Therefore, the 2nd-order proximity and the K-order proximity were proposed to capture the global structural information. Moreover, textual and attributive information of networks increased to such a large scale that cannot be ignored. The recent advanced techniques were designed to extract more accurate features of networks by preserving these additional information. As the sources of this information are enriched, nodes and links in a network are not limited to one single type anymore. The heterogeneity is ubiquitous in real-world networks. Accordingly, based on the basic homogeneous network embedding techniques, novel methods were proposed to model the heterogeneous vertices and edges properties. On the other hand, with the rapid development of networks, vertices evolve further frequently as well as edges and edges' weights. It is important to capture the evolving nature of the networks. Due to the complexity and diversity in different networks, however, techniques that can adequately process dynamic and heterogeneous networks still need to exert more efforts.

In summary, we review existing network embedding techniques using a three-layer categorization framework as shown in Fig. 2. In the first layer, from the perspective of network evolvement, we classify the embedding techniques into two subcategories: network embedding on static networks (not time-varying) and dynamic networks (time-varying). In the second layer, we further classify the static network embedding techniques according to nodes' and edges' types: network embedding on static homogeneous networks and heterogeneous networks. In the third layer, for the static homogeneous networks, in order to preserve the information, we classify the techniques into three groups: network embedding for unstructured data, network embedding with structural information and network embedding with additional information. For the static heterogeneous networks, according to the model they based on, we classify the corresponding techniques into two groups: embedded natural language model and embedded neural network.

4. Network embedding on static networks

4.1. Network embedding on static homogeneous networks

Static homogeneous networks, which are simplified forms of many real-world networks, draw sustaining attention in the area of network embedding. In this category, we concern about two sides complexity of static homogeneous networks: the first is the structural complexity and the second is the additional information complexity. Structural complexity is caused by edges

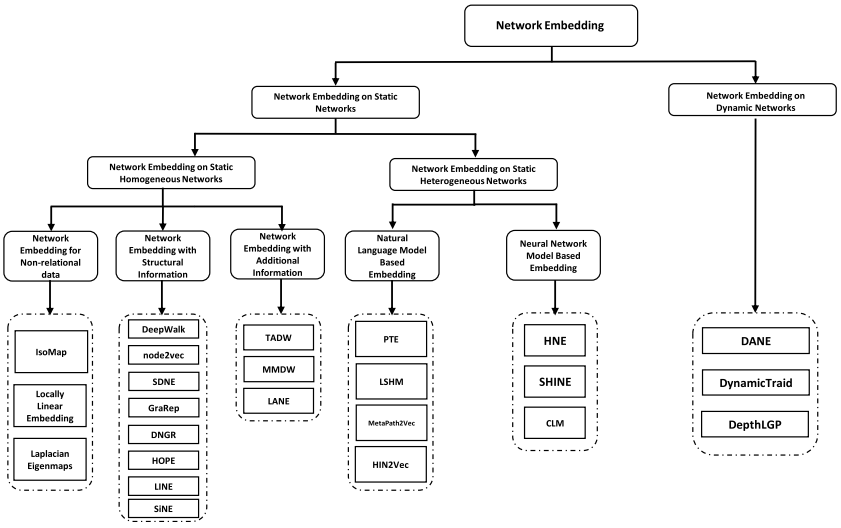

Fig. 2. The three-layer categorization framework.

which can be directed/undirected and weighted/unweighted in static homogeneous networks. Additional information, such as label information and attributive information also pose great challenges to network embedding.

4.1.1. Network embedding for unstructured data

In this category, three earlier embedding methods are introduced. Based on the manifold hypothesis, they represent high dimensional real-world data with low intrinsic essence [37]. These methods transform the unstructured data into a network, and then dimension reduction techniques are adopted to extract features hidden in the data.

Isomap Lin et al. [37] proposed the Isomap aiming to map the high-dimensional data into a low-dimensional space. The algorithm adopts the $K-$nearest neighbors method to represent relationships between the input entity and its neighbor nodes. Then, the shortest path between vertex i and vertex j denoted by $d_G(i,j)$ can be computed to reflect the geodesic distance between two entities in high-dimensional space. After that, classical multidimensional scaling (MDS) [38] is applied on the shortest path matrix $D_G = \{d_G(i,j)\}$ to construct a matrix in the lower Euclidean dimensional space. Isomap tries to minimize the cost function as follows:

$$\sum_{i=1}^{N}\sum_{j=1}^{N}(d_G(i,j) - \|e_i - e_j\|)^2 \tag{3}$$

where N is the number of vertices in the network.

Locally Linear Embedding The deficiency of Isomap is that it needs to search for a global optimal solution, which causes high computational cost. Roweis et al. [15] assumed that entities in input data can be approximated linearly by the $K-$nearest neighbors. Based on this assumption, they proposed the Locally Linear Embedding. In Locally Linear Embedding (LLE), each vertex in the constructed graph can be reconstructed from its neighbors:

$$\min_{W}\sum_{i}\|a_i - \sum_{j} W_{ij}a_j\|^2 \tag{4}$$

where a_i represents the current vertex i and the weight W_{ij} linearly summarize the contribution of a_i's neighbors, a_j. After obtaining the high-dimensional weight W_{ij}, the next step is to learn representations in a low-dimensional space by minimizing the cost function as follows:

$$\min_{\mathbf{e}}\sum_{i}\|e_i - \sum_{j} W_{ij}e_j\|^2. \tag{5}$$

Laplacian Eigenmaps BelKin et al. [16] proposed a method called Laplacian Eigenmaps (LE). This method aims to map input data into a low-dimensional space in a geometrical way. The graph constructed by the algorithm can be divided into two steps: the first step is to build relations between two entities by using the ϵ-neighborhoods or the $K-$nearest neighbors. Secondly, the heat kernel [39] is adopted to represent the weight W_{ij} of edge. Specifically, the Laplacian matrix is utilized to transform the dimensionality reduction problem into the following objective function

$$\begin{aligned}&\sum_{i,j}\|e_i - e_j\|^2 W_{ij} = tr(E^T LE)\\ &s.t.\ \ E^T DE = I, E^T DI = \mathbf{0}\end{aligned} \tag{6}$$

where E is the representation matrix and $\mathbf{0}$ is an eigenvector with eigenvalue 0. In Eq. (6), $L = D - W$, D is a diagonal matrix whose elements are column or row sums of weights. The matrix D preserves the degree of each node in the constructed graph. e_i corresponds to the *ith* row vector in E. Furthermore, the $E^T DE = I$ is a constraint for preventing an arbitrary scaling factor from affecting the embedding output and another constraint $E^T DI = \mathbf{0}$ aims to eliminate the trivial solution.

In summary, the previous methods mentioned in this part regard the network as a form to model unstructured data in the real world. The embedding process equals to matrix factorization. With the emergence of naturally-formed networks, lots of network embedding methods are devoted to extract effective features from sparse, large-scale networks in the real world to support the down-steaming network analysis tasks.

4.1.2. Network embedding with structural information

In this subcategory, according to structural complexity mentioned before, we introduce eight network embedding methods. DeepWalk, Node2Vec, SDNE and GapRep are designed for the simplest network which is undirected and unweighted. DNGR designs a novel sampling strategy on weighted networks. LINE devotes itself to solving embedding problem on the directed and weighted network. SiNE aims to finish the embedding on the signed network and HOPE considers the asymmetric transitivity property in the network which can be seen as a network-specific embedding method.

DeepWalk Perozzi et al. [1] find that word frequency in natural language and degree distribution of network both follow

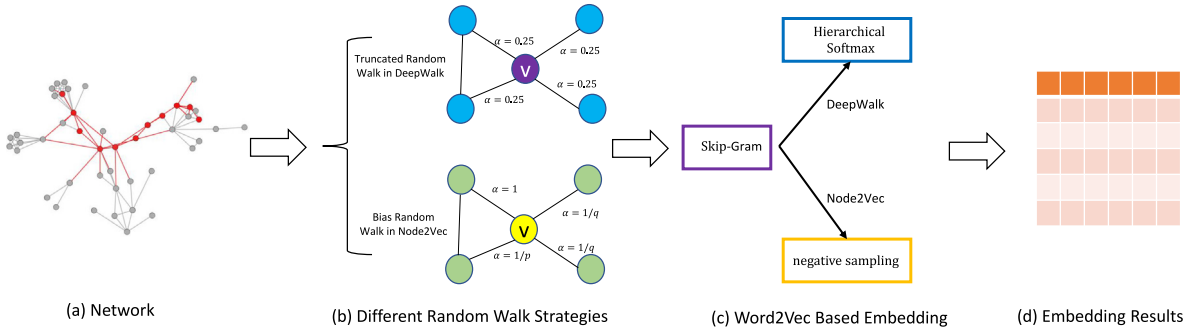

Fig. 3. The framework of DeepWalk and Node2Vec.

power laws. Inspired by this similarity, they proposed DeepWalk. The major contribution of DeepWalk is using a natural language model to represent the community structure in networks. The algorithm utilizes short random walks to extract information from a network by generating a sequence of vertices $s = \{v_1, v_2, \ldots, v_s\}$ corresponding to a sentence in natural language. Specifically, they adopt the SkipGram to learn the representation of a vertex by maximizing the co-occurrence probability with other vertices appearing on the same short random sequence. The equation of the SkipGram is defined as follows:

$$\max_{e_i} Pr(\{v_{i-w}, \ldots, v_{i+w}\} v_i | e_i) \tag{7}$$

where w is the window size in SkipGram Model and $\{v_{i-w}, \ldots, v_{i+w}\}$ is the 'context' nodes set of v_i generated by short random walk. e_i corresponds to the embedding output of the node i. Finally, DeepWalk utilizes hierarchical softmax [40] to reduce computational complexity by transforming the nodes into a huffman tree [41]. Thus, by maximizing the probability from v_i to v_j where $\{j | j \in (i-w, i+w), j \neq i\}$ in a huffman tree, the Eq. (7) changes into:

$$\max_{e_i} Pr(v_k | e_i) = \Pi_{l=1}^{\lceil \log |V| \rceil} Pr(b_l | e_i) \tag{8}$$

where b_l corresponds to a sequence of tree nodes $(b_0, b_1, \ldots, b_{\log |V|})$. From the Eq. (8), it is obvious that the hierarchical softmax function reduces the computational complexity of DeepWalk from $O(|V|)$ to $O(\log |V|)$. The overview of DeepWalk is shown in Fig. 3.

Node2Vec Unlike DeepWalk sampling nodes by performing unbiased random walks on networks, Grover et al. [42] defined a flexible notion of a given node's neighborhoods during embedding process. According to Node2Vec, neighborhoods are not only directed link nodes, but also structural equivalence nodes. For example, two hub nodes in different communities may be far apart even they play similar structural role in the same network. The biased random walk generates neighborhoods of a node by smoothly interpolating BFS and DFS search strategies aiming at finding directed neighborhoods and structural equivalence neighborhoods.

Similar to the DeepWalk, Node2Vec turns the embedding problem into maximizing the probability of finding the co-occurrence neighbor vertices $N(v_i)$ by utilizing the SkipGram. Then, the embedding vector can be defined as follows:

$$\max_{e_i} \sum_{v_i \in V} \log Pr(N(v_i) | e_i) \tag{9}$$

Furthermore, the negative sampling method [40] is leveraged to solve the high computational complexity by regarding the neighborhood nodes of v_i as 'negative sampling'. The overview of Node2Vec is shown in Fig. 3.

SDNE The Deepwalk and Node2vec methods utilize the Word2Vec as their basic embedding model which is a log-linear natural language processing model [43]. To capture high nonlinear structure, numerous methods are based on the shadow model or kernel techniques [17]. Compared with deep learning frameworks, most shadow models or kernel techniques based methods cannot obtain a powerful embedding output [44]. Accordingly, Wang et al. [45] designed a semi-supervised deep model to measure the 1st-order and the 2nd-order proximity in the network. Based on the 1st-order proximity, the local neighboring information among nodes can be preserved. Furthermore, the 2nd-order proximity in SDNE is used to solve the sparsity problem caused by the 1st-order proximity. To model the 2nd-order proximity, SDNE designs an unsupervised component to reconstruct the neighboring relations between nodes. An autoencoder algorithm is used to represent the 2nd-order proximity as follows:

$$\begin{aligned} y_i^{(1)} &= \sigma(W^{(1)} e_i^{(1)} + b^{(1)}) \\ y_i^{(k)} &= \sigma(W^{(k)} e_i^{(k-1)} + b^{(k)}), k = 2, \ldots, K \end{aligned} \tag{10}$$

where σ is the sigmoid function playing an active function. The hidden embedding output of each layer is $y_i^{(j)}$. $e_i^{(k)}$ is the kth layer embedding output and the goal of the autoencoder is to minimize the 'distance' from input to output as follows:

$$L_{2nd} = \Sigma_{i=1}^{n} \|(\hat{e}_i - e_i) \odot b_i\|_2^2 \tag{11}$$

where $\hat{e}_i$ is the final embedding output. $\odot$ corresponds to Hadamard product [46] and b_i is the penalty factor for adding more penalty to reconstruction error.

On the other hand, the 1st-order proximity incorporates Laplacian Eigenmaps to finish the constructing work with the loss function as follows:

$$L_{1st} = \Sigma_{i,j=1}^{n} \|s_{i,j} e_i - e_j\|_2^2 \tag{12}$$

where $s_{i,j}$ is a parameter adding penalty to similar vertices along the path. Finally, SDNE combines the 1st-order with the 2nd-order proximity and then optimizes this combination using the backpropagation method.

GraRep Deepwalk, Node2vec and SDNE design different strategies to preserve the 2nd-order proximity structural information of networks. However, the 2nd-order proximity is only a two-step measurement of nodes reflecting local structural information. Cao et al. [32] consider the high-order proximity to extract global information of network by building a K-step probability transition matrix of network: $\mathbf{A}^k = \underbrace{\mathbf{A} \ldots \mathbf{A}}_{k}, \mathbf{A} = D^{-1}M$. D is a degree matrix and M is an adjacency matrix. The element A_{ij}^k reflects the probability of a K-step path from node i to node j. Inspired by DeepWalk, SkipGram is utilized to define the loss function

combined with the negative sampling method as follows:

$$L_k(i) = (\sum_j p_k(j|i) \log \sigma(e_j^T e_i)) + \lambda \mathbb{E}_{j' \sim p_k(V)}[\log \sigma(-e_i^T e_j')] \quad (13)$$

where $p_k(j|i)$ is the K-step transition probability equaling to A_{ij}. $\sigma(\cdot)$ is a sigmoid function and the negative sample j' follows the distribution $p_k(V)$. Its expectation can be measured by $\mathbb{E}_{j' \sim p_k(V)}[\cdot]$. The first half of Eq. (13) is the cross entropy [47] corresponding to the 'score' of positive samples and the second half is the 'score' of negative samples. Therefore, the optimization goal of this formula is to maximize the first half and minimize the second half. In the GraRep, the optimization problem is solved by a classical matrix factorization technique—SVD [48]. Fig. 4 shows the framework of GraRep.

DNGR Sampling methods such as truncated random walk can be performed well on the unweighted networks. However, these methods cannot be applied directly on the weighted networks. Therefore, improved sampling methods based on the random walk is proposed such as Node2Vec. Cao et al. [49] design a random surfing model called DNGR. Inspired by PageRank [50], DNGR transforms weighted networks into a probabilistic co-occurrence matrix. Then, positive point-wise mutual information (PPMI) [51] is adopted to remove stop words in natural language, and the SVD technique [48] is utilized to reduce the high dimensional PPMI matrix. Finally, in order to obtain the embedding output, DNGR transforms PPMI matrix into a stacked denoising autoencoder by using dimension reduction techniques.

HOPE In directed networks, one critical property is the asymmetric transitivity. It means that two different nodes on a path (the distance is more than one-hop) will have a higher propensity for sharing a directed link whose direction is the same as the path. Furthermore, this propensity is consistent with the number of the paths from v_i to v_j and the length of these paths.

To preserve this high-order proximity, Ou et al. [2] introduce a source vector e^s and a target vector e^t to represent embedding vector where their inner product measures the proximity from node i to node j. The general formulation of high-order proximity is proposed after concluding four frequently-used measurements including Katz Index [52], Rooted PageRank [53], Common Neighbors [53] and Adamic-Adar [54]. These four measurements focus on the global proximity (Katz Index, Rooted PageRank) and the local proximity (Common Neighbors, Adamic-Adar) respectively. The general formulation is as follows:

$$S = M_g^{-1} \cdot M_l \quad (14)$$

where M_g^{-1} concerns the global asymmetric transitivity and M_l constructs the transiting range of the subgraph. The K-order proximity matrix is represented by S. Finally, SVD is utilized to solve the matrix factorization problem and the embedding output can be obtained.

SiNE In weighted networks, weight may be positive or negative. In various application scenarios, negative links play a more important role than positive ones in improving the prediction performance [55] or recommendation performance [56]. Wang et al. [57] concentrate on embedding social media networks by using a famous social theory—the structural balance theory [58, 59]. This theory assumes that a user should have closer intimacy with friends (positive links) than 'foes' (negative links) in a signed social network. In SiNE, the major procedure is to transform the node with only positive or negative links into two triplets. The triplets have one positive link and one negative link by adding the virtual nodes. Therefore, the structural balance can be mathematically modeled as:

$$f(e_i, e_j) \geq f(e_i, e_0) + \xi_0 \quad (15)$$

where e_i and e_j are embedding results of v_i and v_j. e_0 corresponds to a virtual node and ξ_0 is a parameter used to regulate the distance of two similarities. Finally, a two-layer deep network is proposed as shown in Eq. (10). In order to obtain the embedding output, SiNE uses the backpropagation to optimize this deep model.

LINE For arbitrary structural types of networks: undirected, directed, and/or weighted, Tang et al. [20] propose LINE which can easily be applied to various structural types of networks with millions of nodes. It is worth noting that the 1st-order proximity cannot be fully observed in many networks, leading to an unsatisfactory result in preserving the global information of a network. The key innovation of LINE is that it combines the 1st-order and the 2nd-order proximity to retain the structural information of networks. The 2nd-order proximity measures the similarity of two nodes by counting the common neighborhoods of them. The KL-divergence [60] of two probability distributions is utilized to generate the objective function for computing the 'divergence' between the joint probability of embedding pairs and the empirical probability of original pairs. The 1st-order proximity's objective function is:

$$\min O_1 = -KL(\hat{p_1}(\cdot,\cdot), p_1(\cdot,\cdot)) \quad (16)$$

where $\hat{p_1}(v_i, v_j)$ is the empirical probability equaling to $\frac{W_{ij}}{\sum_{(i,j)\in E} W_{ij}}$ and $p_1(v_i, v_j)$ is the joint probability equaling to $\frac{1}{1+\exp(-\langle e_i^T \cdot e_j \rangle)}$. $e_i \in R^d$ is the low-dimensional embedding output of node v_i. The 1st-order proximity preserves neighboring linking relation of node pairs and can only be applied to undirected networks. Similar to the 1st-order proximity, the 2nd-order proximity's objective function is:

$$\min O_2 = -KL(\hat{p_2}(\cdot|\cdot), p_2(\cdot|\cdot)) = -\sum_{(i,j)\in E} W_{ij} \log p_2(v_i, v_j) \quad (17)$$

where $p_2(v_j|v_i)$ equals to $\frac{\exp(e_j'^T \cdot e_i)}{\sum_{k=1}^{|V|} \exp(e_k'^T \cdot e_i)}$. Similar to DeepWalk, this method treats e_j' as the 'context' of other vertices. Thus, $\hat{p_2}(v_j|v_i)$ reflects the probability that the 'context' v_j is generated by v_i. The empirical probability of $\hat{p_2}(v_j|v_i)$ can be represented as $\frac{w_{ij}}{d_i}$ and d_i is the out-degree of node i.

In summary, the first four methods design their embedding methods on the unweighted/undirected network. They pay more attention on preserving structural-proximity information by using different basic strategies. As shown in Figs. 3 and 4, Deepwalk, Node2vec, DNGR utilize random walk based strategy and GraRep adopts matrix factorization based strategy. SDNE and DNGR devise deep neural network [61] to capture the non-linearity of network. Four remaining methods focus on the structural-property information such as weighted/directed and asymmetric information.

4.1.3. Network embedding with additional information

Previous methods mainly focus on network structural information. However, the additional information such as users' interested topics in social networks or text information of nodes in webpage networks depict characteristics of networks from another side. Methods combining structural and additional information perform better in embedding results than just preserving structural information.

TADW Depending on rich text information, Yang et al. [21] propose TADW by utilizing matrix factorization to incorporate text feature matrix into product factor of the decomposition term. Firstly, they prove the intrinsic relationship between DeepWalk and matrix factorization as follows:

$$M_{ij} = \frac{[e_i(A + A^2 + \cdots + A^t)]}{t} \quad (18)$$

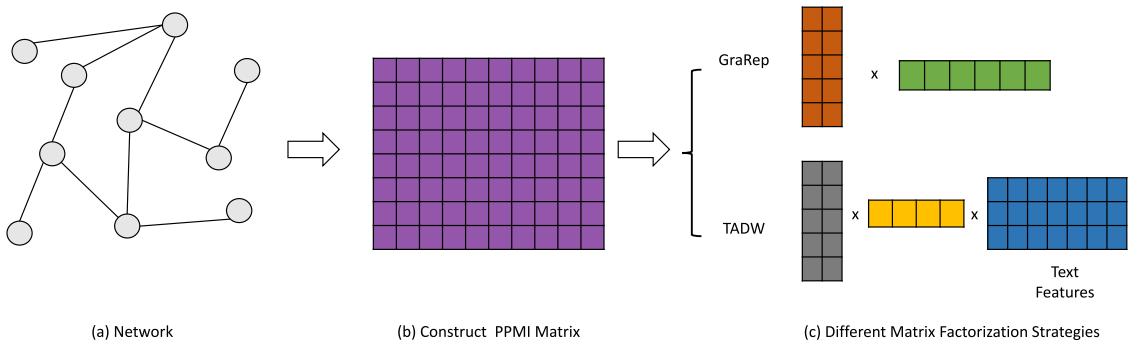

Fig. 4. The framework of GraRep and TADW.

where M_{ij} corresponds to PMI matrix of word-context pair, which is similar to DeepWalk. e_i is an indicator row vector, where the ith member is 1 while others are 0. The element of A is reciprocal of node's degree. Therefore, $e_i A^k$ measures the transition probability from one node to another by k times that can be analogous to the GraRep.

Similar to DeepWalk, the low-rank matrix factorization [62] method and inductive matrix completion [63] are incorporated to construct the loss function. By low-rank matrix factorization, TADW extracts elements contained in the matrix with additional features.

$$\min_{W,H} \|M - W^T HT\|_F^2 + \frac{\lambda}{2}(\|W\|_F^2 + \|H\|_F^2) \tag{19}$$

It assumes that the matrix $M \in \mathbb{R}^{b\times d}$ is approximate to low rank k where $k \ll b, d$. The W and H are matrices that the optimization process aims to find. $\|\cdot\|_F$ means Frobenius norm of matrix and λ is a harmonic factor. Finally, by optimizing the W and HT, the embedding output can be obtained. The framework of TADW can be found at Fig. 4, we can observe the enhancement of TADW in text features incorporating compared with GraRep.

MMDW Inspired by DeepWalk and TADW, Tu et al. [64] designed a 'task-dependence' algorithm with the node classification task. The classical algorithm for classification algorithm SVM [65] is utilized to find the max-margin classifier boundaries by adding the node's label information. Therefore the objective function is defined as follows:

$$\begin{aligned} &\min_{X,Y,W,\xi} L = \min_{X,Y,W,\xi} L_{DW} + \frac{1}{2}\|W\|_2^2 + C\Sigma_{i=1}^{T}\xi_i \\ &s.t \quad el_i{}^T x_i - e_{l_j}^T x_i \geq e_i^j - \xi_i, \forall i, j \end{aligned} \tag{20}$$

where $\min_{X,Y,W,\xi} L_{DW}$ is the objective function in DeepWalk and the remaining part is SVM optimization formula. Biased Gradient is introduced to optimize the parameter X, Y, W, ξ in the Eq. (20).

LANE In the attributive networks, features of nodes and related label information pose challenges to embedding work because of its sparsity and noise [66]. Therefore, Huang et al. [22] design a framework to obtain a unified embedding of labeled attributive networks in two steps. Firstly, they adopt cosine similarity to construct an affinity matrix to measure the pairwise similarity of elements. The Laplacian matrix [16] is introduced to measure the degree of disagreement between structural and attributive properties of nodes. Secondly, the label information is modeled by a Laplacian matrix and the first-step embedding out $\epsilon^{(G)}\epsilon^{(A)}$ is utilized to solve the smooth problem caused by limited label number just like the first step. Consequently, these three embedding outputs are projected into a new common space by obtaining output of correlations reflected by variance of the projected matrix. The final embedding results capture both topology properties and label information in attributive networks.

In summary, these three methods, incorporate attributive information into embedding process in different ways. TADW factorizes the text information matrix of network. MMDW adopts SVM method in constructing the objective function. The node attributive information is imported by the classifier of SVM. Compared with TADW and MMDW, LANE preserves attributive information and labeled information jointly in the embedding process.

4.2. Network embedding on static heterogeneous networks

As mentioned above, static homogeneous network is not prevalent in the real world. A node of network representing an entity, usually contains rich information. This information may be collected from different resources. Therefore, the edge type may not be limited into a single one. Heterogeneous networks, with more than one type of nodes and edges, pose a significant challenge to network embedding in capturing the heterogeneity of nodes and edges.

In this subsection, we introduce some recent proposed network embedding methods aiming to embed the heterogeneous network into a low-dimensional space. According to the model they used, these methods can be categorized into two types: embedded natural language model and embedded neural network model.

4.2.1. Natural language model based embedding

Similar to the DeepWalk, methods in this categorization basically utilize the techniques commonly used in natural language process to capture relationships of nodes and generate the objective function as the form of Bayes.

PTE Tang et al. [67] propose a semi-supervised algorithm for heterogeneous textual networks. They first transform the heterogeneous text network into three bipartite networks: word–word network, word–document network and word–label network. By incorporating the idea of LINE, the 2nd-order proximity is used to generate the objective function. To achieve corresponding heterogeneous network embedding, these three bipartite embedding objective functions are aggregated together as follows:

$$O_{pte} = O_{ww} + O_{wd} + O_{wl} \tag{21}$$

where $O(\cdot)$ shares the same form with Eq. (16). In order to obtain the result of O_{pte}, PTE proposes two possible training method: joint training, pre-training and fine-training. The major difference between these two methods is the way to process unlabeled data and labeled data: joint training trains the model with unlabeled data and labeled data together. The pre-training and fine-training trains the model with unlabeled data at first and then uses labeled data to fine-tune the embedding output.

In brief, PTE embeds heterogeneous text network with word co-occurrences by dividing a heterogeneous network into different homogeneous networks. It projects word, document and

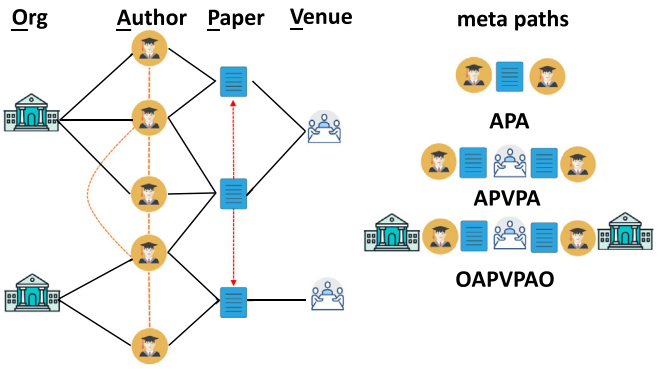

Fig. 5. An illustrative example of metapath in academic network.
Source: The figure is adapted from [70].

label information into a common space by introducing specific classification task.

LSHM Similar to PTE, some embedding methods designing for heterogeneous networks mainly focus on mapping one heterogeneous network into different homogeneous networks and then classical network embedding methods can be used. However, the loss dependent information between different types of data cannot be ignored especially for a large-scale heterogeneous network. To solve this problem in heterogeneous networks, Jacob et al. [68] propose an algorithm to exploit dependencies between different types of nodes. Similar to MMDW, this proposed method designs a transductive classification function $f_\theta^{t_i}$ to predict a node's label for different types of nodes and adds it to the loss function as follows:

$$\begin{aligned} L(z,\theta) &= \Sigma_{i=1}^{l} \Delta(f_\theta^{t_i}(z_i), y_i) \\ &+ \lambda \Sigma_{i,j/w_{i,j}\neq 0} w_{i,j} \|e_i - e_j\|^2 \end{aligned} \tag{22}$$

where $\Delta(\cdot)$ denotes the loss function for measuring the accuracy of label's classifier and l is the number of labeled nodes. The second half of Eq. (22) is a smoothness constraint making two nodes i and j close in embedding space as their weight $w_{i,j}$ reaching to a threshold. Finally, SGD [69] method is utilized to obtain the embedding output. Following above steps, LSHM projects different types of nodes into a common space while preserving the dependencies between them.

MetaPath2Vec In PTE, the content is decomposed into words, and then, word–document network is constructed. In LSHM, the dependencies between different nodes are simplified through a classification predicator. Both of them simplify the heterogeneity in networks. Different with them, Swami et al. [71] utilize metapath [70] to measure the proximity between different types of nodes. A biased metapath-based random walk is designed to explore heterogeneous networks. The idea of Word2Vec [40] is introduced to generate the embedding output as well as the DeepWalk. The metapath [70] can be defined as $V_1 \xrightarrow{R_1} V_2 \xrightarrow{R_2} \ldots V_t \xrightarrow{R_t} V_{t+1} \ldots \xrightarrow{R_{l-1}} V_l$, where $R = R_1 \circ R_2 \circ \cdots \circ R_{l-1}$ defines a kind of operation formulating relations of node types. Therefore, the transition probability p at the *ith* step can be computed as follows:

$$p(v^{i+1}|v_t^i, \mathcal{P}) = \begin{cases} \dfrac{1}{|N_{t+1}(v_t^i)|} & (v^{i+1}, v_t^i) \in E, \phi(v^{i+1}) \doteq t+1 \\ 0 & (v^{i+1}, v_t^i) \in E, \phi(v^{i+1}) \neq t+1 \\ 0 & (v^{i+1}, v_t^i) \notin E \end{cases} \tag{23}$$

where $v_t^i \in V_t$ and $N_{t+1}(v_t^i)$ represent the V_{t+1} type of neighborhoods for node i and $\mathcal{P}$ corresponds to the metapath. Based on Eq. (23), we can conclude that the metapath-based random walk prefers sampling semantic relationships to structural relationships.

Furthermore, one deficiency of the metapath2vec is that it treats all types of nodes as the negative samples in the softmax process. Therefore, the metapath2vec++ is proposed to add types' information to the softmax function. Then, the negative sampling method is aggregated into the objective function. Finally, the metapath2vec++ makes the number of embedding results equivalent to the number of elements contained in each type. The framework of metapth2vec and metapath2vec++ can be found in Fig. 5.

Hin2vec Similar to metapath2vec, Fu et al. [72] utilizes metapath to simple the nodes from heterogeneous networks. Different with metapath2vec, instead of embedding link-type information in the negative sampling phase in metapath2vec, the proposed Hin2vec treats link-type as the specific information in designing their embedding strategy. Hin2vec consists of two phase: the first phase is to prepare the training data for obtaining the proper training data based on the metapath random walk and negative sampling; the second phase is an embedding procedure aiming at learning representations of nodes' vectors based on predicting the link-type of two nodes. The objective function can be defined as follows:

$$\begin{aligned} O_{x,y,r}(x,y,r) &= \begin{cases} P(r|x,y), & L(x,y,r)=1 \\ 1-P(r|x,y), & L(x,y,r)=0 \end{cases} \\ \log O_{x,y,r}(x,y,r) &= L(x,y,r)\log P(r|x,y) \\ &+ [1-L(x,y,r)]\log[1-P(r|x,y)] \\ P(r|x,y) &= sigmoid(\sum W_x'\vec{x} \odot W_y'\vec{y} \odot f_{01}(W_R'\vec{r})) \end{aligned} \tag{24}$$

where x, y correspond to two nodes and r is the type of relationship between x and y. $W(\cdot)$ is the projection matrix which can transform the vector into a latent representation space. One deficiency of random walk technique is that it can only find the positive data that causes problems during training without negative samples. To solve this problem, Hin2vec designs a strategy by randomly replacing x or y with other types of data and then filters out the erroneously generated positive samples.

In summary, PTE and LSHM extract the node information to construct heterogeneous networks. MathPath2Vec and Hin2vec pay more attention to intrinsical heterogeneity of network. Therefore, the heterogeneous network sampling method, metapath random walk are incorporated into embedding. In heterogeneous network embedding, metapath is an effective and important technique to sample nodes.

4.2.2. Neural network model based embedding

Similar to SDNE and DNGR, deep neural network model has some advantages in handling the non-linear embedding. In this subsection, we introduce three methods based on deep-model to solve the embedding challenges in the heterogeneous networks.

HNE Different from word/document heterogeneous network in PTE, multimedia networks (e.g., news networks) pose a larger challenge to embed different types of nodes into a unified space. The reason behind it is that the intrinsic difference between vectorized nodes (e.g., text documents) and tensor-based multimedia nodes (e.g., image or video). Chang et al. [73] combine Deep Neural Networks (DNN) [74,75] with Convolutional Neural Networks (CNN) [76,77] to finish the embedding task on the heterogeneous networks.

They use two linear transformation matrices to convert two types of samples (image, text) into a common latent space denoted by $\mathbf{U} \in \mathbb{R}^{d_I' \times r}$ and $\mathbf{V} \in \mathbb{R}^{d_T \times r}$. d_I' and d_T correspond to the

number of dimension after embedding these two kinds of nodes. Therefore, the three types of the relationships transformation can be defined as follows:

$$\begin{aligned} s(I_i, I_j) &= (U^T I_i)^T U^T I_j = I_i^T M_{II} I_j \\ s(T_i, T_j) &= (V^T T_i)^T V^T T_j = T_i^T M_{TT} T_j \\ s(I_i, T_j) &= (U^T I_i)^T V^T T_j = I_i^T M_{IT} T_j \end{aligned} \tag{25}$$

and then the loss function is formulated as follows:

$$L(I_i, I_j) = \log(1 + \exp(-A_{i,j} d(I_i, I_j))) \tag{26}$$

where $d(\cdot)$ is a link information encoding function and $A_{i,j}$ is an adjacency matrix. I and T correspond to 'Image' and 'Text' samples which are transformed to a uniform latent space. The same as Eq. (26), the image nodes' loss function, the text–text and image–text's loss function can be defined as $L(T_i, T_j)$ and $L(I_i, T_j)$ respectively. The final loss function of HNE combines three loss functions and introduces the corresponding balance parameter λ_1, λ_2 and λ_3.

Finally, in order to extract features from image and text data effectively, CNN and DNN techniques are aggregated into this deep architectures to obtain the unify embedding output of heterogeneous multimedia networks.

SHINE Wang et al. [78] propose an end-to-end network embedding framework for signed heterogeneous networks which can be seen as an extension of SiNE. It is designed to obtain the representations of users from heterogeneous social networks and predict the latent sentiment links. SHINE first collects various kinds of data including Weibo Tweets, Social Relation, Profile of Ordinary Users [79] and Profile of Celebrities. Then, it designs an index to reflect the users' sentiment towards celebrities in tweets called the SO (sentiment orientation) scores:

$$SO(word) = PMI(word, pos) - PMI(word, neg) \tag{27}$$

It reflects the difference of two point-wise mutual information [51]. Here $PMI(x, y)$ equals to $\log \frac{p(x,y)}{p_x p_y}$. The original heterogeneous network is divided into three homogeneous networks: sentiment network, social network and profile network. Specially, three autoencoder algorithms are introduced to solve the embedding challenge with a 6-layer neural network, the hidden embedding can be formulated as follows:

$$e_i^k = \sigma(W_s^k e_i^{k-1} + b_s^k), k = 1, 2, \ldots, K_s \tag{28}$$

and the objective function is defined by using the 1st-order proximity as well as SDNE:

$$\mathcal{L}_s = \sum_{i \in V} \|(x_i - x_i') \odot l_i\|_2^2 \tag{29}$$

Similar to SDNE, $\odot$ denotes Hadamard product [46] which can impose more penalty to reconstruction errors of non-zero elements. Finally, three available aggregation functions: *Summation* [80], *Max pooling* [81] and *Concatenation* [20] can be utilized to combine three network embeddings together and the *Inner product* [73], *Euclidean distance* and *Logistic regression* [1] are introduced to calculate the predicted sentiment $\bar{s}_{ij}$ by utilizing the embedding results of user i and user j.

Curriculum Learning Method Qu et al. [82] propose a method which aims at learning an optimal sequence of edges with different types based on the deep reinforcement learning. The heterogeneous star networks, similar to the knowledge graph [83,84] or the attributive network, consist of center nodes with multiple attributive nodes' types and edges' types. Therefore, the embedding sequence of edges have different impacts on embedding output due to the corresponding semantic descriptions. For example, in a bibliography star network, the venue of a paper reflects the shadow semantic meaning while keywords or references reflect the detailed and deep semantic meanings of a node. In essence, it is a sequence decision procedure modeled as a Markov decision process. The key idea of this framework is to maximize the type-learning-sequence decision making scores $Q(s, a)$. It equals to optimizing the expected cumulative reward after taking action a from the state s. In order to obtain the optimal result, this framework based on the idea of reinforcement learning designs two modules: the *planning* module and the *learning* module focusing on effectiveness and efficiency respectively. The benefit of this strategy is that it can avoid falling into the trap of locally optimal decision.

From the current state s, the *planning* module uses the Monte-Carlo search tree to find a number of actions to explore until it reaches an unvisited state. Motivated by the UCT algorithm [85], this module selects the next action based on the following term:

$$\begin{aligned} a = \arg\max_a & \frac{Q_p(s, a)N(s, a)}{N(s, a) + 1} + \frac{Q_l(s, a)N(s, a)}{N(s, a) + 1} \\ & + \lambda \sqrt{\frac{lnN(s)}{N(s, a) + 1}} \end{aligned} \tag{30}$$

where $Q_p(s, a)$ and $Q_l(s, a)$ represent the *planning* module and the *learning* module's Q value respectively. $N(s, a)$ is a visiting count and $N(s) = \sum_a N(s, a)$ is the total visiting number of the state s. λ is a parameter used to balance the search strategies between the exploitation and the exploration. From Eq. (30) we can find that the *planing* module encourages the decision-making procedure to get a larger Q value in the past. The following step is updating the Q_p according to the temporal difference learning method [86]:

$$\begin{aligned} Q_p(s_i, a_i) \leftarrow & Q_p(s_i, a_i) \\ & + \alpha[r_i + Q_p(s_{i+1}, a_{i+1}) - Q_p(s_i, a_i)] \end{aligned} \tag{31}$$

where $\alpha = \frac{1}{N(s_i, a_i)}$ is the learning rate.

Contrary to the *planing* module which is a forward process, the *learning* module makes a decision based on the experience. Naturally, the idea of deep learning can be utilized in this phase to receive the *planing* module data as the training data and learn the parameter of the deep neural network. The updating process can be formulated as follows:

$$w_l \leftarrow w_l + \alpha[r_i + Q_l(s_{i+1}, a_{i+1}) - Q_l(s_i, a_i)] \bigtriangledown_{w_l} Q_l(s_i, a_i) \tag{32}$$

where α is the learning rate updated by the RMSProp algorithm [87] and w_l is the parameter set of the neural network.

Finally, Q_p and Q_l are aggregated as follows:

$$\hat{Q}(s, a) = \frac{Q_p(s, a)N(s, a)}{N(s, a) + 1} + \frac{Q_l(s, a)N(s, a)}{N(s, a) + 1} \tag{33}$$

Similar to LINE, by getting an action a, it can be used to calculate the divergence between the neighborhood distribution $p(\cdot|v)$ and the empirical neighborhood distribution $\hat{p}(u|v)$ to obtain the embedding of each node.

The neural network model has two dominant superiorities: one is the excellent performance on the non-linear learning tasks and the other is that it can be designed for an end-to-end network embedding especially in the heterogeneous networks. In this subsection, three methods are introduced for specific heterogeneous network: multimedia network heterogeneous network, signed heterogeneous network and star heterogeneous network. Similar to PTE, HNE and SHINE divide the heterogeneous network into homogeneous networks and then combine embedding results. Curriculum Learning Method transforms the edge type embedding into a sequence decision problem.

5. Network embedding on dynamic networks

Capturing the pattern of network evolvement is the pivotal approach to better understand the essence of a network [88]. Therefore, network embedding aiming at tackling the dynamic nature of network is always an important research direction [89]. However, related works are scarce due to its complexity. In this section, three recent dynamic network embedding methods capturing different dynamic features of networks are introduced.

5.1. DANE

The framework of DANE [13] consists of two parts: data preprocessing and matrix perturbation. During the data preprocessing process, noisy and incomplete data are filtered out. Then, matrix perturbation is applied on the processed data when the network structure and attributes alter.

Similar to the static homogeneous network algorithms mentioned before, this method designs a basic off-line (static) embedding method for preserving nodes' pairwise relationships and nodes' attributions. In this part, they first model the network and attribution as two matrices. Then, by utilizing the spectral techniques, they transform these two matrices into a Laplacian form. Therefore, according to matrix factorization theory, noise information can be reduced by getting the top-n eigenvectors. Afterwards, based on the representation of different attributes, a consistent representation is obtained, which is realized by two projection matrices as follows:

$$\begin{aligned}&\max_{P_A^{(t)},P_X^{(t)}} P_A^{(t)'}Y_A^{(t)'}Y_A^{(t)}P_A^{(t)} + P_A^{(t)'}Y_A^{(t)'}Y_X^{(t)}P_X^{(t)}\\&+ P_X^{(t)'}Y_X^{(t)'}Y_A^{(t)}P_A^{(t)} + P_X^{(t)'}Y_X^{(t)'}Y_X^{(t)}P_X^{(t)}\\&s.t.\ \ P_A^{(t)'}Y_A^{(t)'}Y_A^{(t)}P_A^{(t)} + P_X^{(t)'}Y_X^{(t)'}Y_X^{(t)}P_X^{(t)} = 1\end{aligned} \tag{34}$$

where $Y_{(\cdot)}$ is the embedding vector and $P_{(\cdot)}$ is the projection vector. A and X represent the structure and attribution respectively. This constrained optimization problem can be solved by adding a Lagrange multiplier. The final consensus embedding form can be represented as $E^{(t)} = [E_A^{(t)}, E_X^{(t)}] \times P^{(t)}$.

Considering the real-world networks, especially the attributive networks [90], often evolve smoothly between two time intervals, $\triangle A$ and $\triangle X$ are used to represent the perturbation of network structure and node attribution respectively from t to $t+1$. In the light of this, embedding updating online can be formulated by the change of two matrices: the diagonal matrix and the Laplacian matrix:

$$\begin{aligned}&D_A^{(t+1)} = D_A^{(t)} + \triangle D_A, L_A^{(t+1)} = L_A^{(t)} + \triangle L_A,\\&D_X^{(t+1)} = D_X^{(t)} + \triangle D_X, L_X^{(t+1)} = L_X^{(t)} + \triangle L_X.\end{aligned} \tag{35}$$

Taking the structural part as an example, the updating problem can be modeled as:

$$(L_A^{(t)} + \triangle L_A)(a + \triangle a) = (\delta + \triangle\delta)(D_A^{(t)} + \triangle D_A)(a + \triangle a) \tag{36}$$

where a corresponds to an eigenvector of structural information matrix A. Based on the matrix perturbation theory, the updated eigenvalue $\triangle\lambda_i$ is:

$$\triangle\lambda_i = a_i'\triangle L_A a_i - \lambda_i a_i'\triangle D_A a_i \tag{37}$$

where a_i' is a vector conforming $a_i' D_A^{(t)} a_i = 1$ and $a_j' D_A^{(t)} a_j = 0 (i \neq j)$. The corresponding change of eigenvector $\triangle a_i$ is:

$$\begin{aligned}\triangle a_i = &-\frac{1}{2}a_i'\triangle D_A a_i a_i\\&+ \sum_{j\doteq 2, j\neq i}^{k+1}\left(\frac{a_j'\triangle L_A a_i - \lambda_i a_j'\triangle D_A a_i}{\lambda_i - \lambda_j}\right)a_j\end{aligned} \tag{38}$$

Finally, by updating this two matrices, the consequent embedding output can be naturally obtained.

DANE will show a more efficient performance in fact than repeatedly refreshing an off-line embedding method when the network shifted, due to the sparsity of real-world networks adjacent matrices.

5.2. Dynamic triad

DANE assumes a smooth change in dynamic networks. However, sharp evolvement is very common in networks. How to preserve evolutionary patterns effectively is challenging to dynamic network embedding. To solve this, Zhou et al. [91] introduce the *triad* as one of the basic units of networks. It contains three vertices, which is similar to the idea of motif [92,93].

The *triad* consists of two types: *closed triad* and *open triad*. These two types correspond to a 3-vertex complete graph and a bipartite graph with only two edges respectively. The mathematical description of evaluation from *open* to *closed* is the key to capture properties of dynamic networks. Therefore, the probability that an open triad (v_i, v_j, v_k) evolves into a closed one can be formulated as follows:

$$P_{tr}^t(i,j,k) = \frac{1}{1+\exp(-\langle\theta, x_{ijk}^t\rangle)} \tag{39}$$

where θ is the *social strategy parameter*. x_{ijk}^t reflects the affinity of v_k with other two nodes.

Furthermore, as well as the 2nd-proximity order, the evaluation procedure may be interrupted by the outsider common friends' nodes. Therefore, DynamicTraid defines the probability of creating a new link e_{ij} as $P_{tr_+}^t(i,j)$ while the opposite of probability as $P_{tr_-}^t(i,j)$. Finally, by combining the $P_{tr_+}^t(i,j)$ with $P_{tr_-}^t(i,j)$, the objective function for measuring triad closure can be defined as follows:

$$\begin{aligned}L_{tr}^t = &-\sum_{(i,j)\in S_+^t}\log P_{tr_+}^t(i,j)\\&-\sum_{(i,j)\in S_-^t}\log P_{tr_-}^t(i,j)\end{aligned} \tag{40}$$

where $S_+^t = \{(i,j)|e_{ij} \notin E^t \wedge e_{ij} \in E^{t+1}\}$ is the set indicating the successfully created cases at $t+1$ and $S_-^t = \{(i,j)|e_{ij} \notin E^t \wedge e_{ij} \notin E^{t+1}\}$ is the set representing the failure cases.

In order to make the DynamicTraid algorithm more robust, the commonly accepted assumptions: social homophily and temporal smoothness theories are incorporated to model the *triad*. The social homophily are inclined to believe that the linked nodes should be similar in the embedding space and the temporal smoothness provides an assumption that most existing networks will evolve in a smooth way. The final form of the objective function at T time can be defined as:

$$\arg\min_{u_i^t,\theta}\sum_{t=1}^{T} L_{sh}^t + \beta_0 L_{tr}^t + \beta_1 L_{smooth}^t \tag{41}$$

where L_{sh}^t and L_{smooth}^t correspond to the social homophily and the temporal smoothness respectively.

5.3. DepthLGP

Different from DynamicTraid only concerning the evaluation of edges in a 3-vertex subgraph, Ma et al. [7] aim to propose a method to infer embeddings for out-of-sample nodes. Inspired by non-parametric probabilistic modeling and deep learning, this method utilizes a high-order Laplacian Gaussian procedure to

preserve network properties which satisfies the fast and scalable inference simultaneously.

The out-of-sample nodes refer to nodes arrived after learning. To model out-of-sample nodes in a dynamic network, they first defines $h(v) = [h_1(v), h_2(v), \ldots, h_s(v)]^T$ that can independently encode each node. And then, each $h_k(\cdot)$ can be modeled by a kernel function that computes the similarity between nodes. Hence, kernel sets can be represented as a kernel matrix. Considering the 1st-order and the 2nd-order proximity, Laplacian matrix is utilized to represent the Euclidean distance between the out-of-sample updating matrix and the original matrix. Furthermore, DepthLGP assumes that the element of kernel matrix $h_k(\cdot)$ is in line with a zero-mean Gaussian process (GP) [94–96].

In the prediction part, DepthLGP aims to generate a training procedure on the network by embedding function $\mathbf{f}(v)$ before new nodes have arrived. Therefore, prediction procedure can be formulated as follows:

$$p(\mathbf{f}(v) : v \in V^* | \mathbf{h}(v) \in V) \tag{42}$$

In the training part, DepthLGP first samples some subgraphs from networks. Part of nodes in these subgraphs are treated as the out-of-sample nodes. The optimization purpose is to minimize the empirical risk on these training samples.

In summary, these three methods focus on the different dynamic characteristics. The DANE uses the traditional way to solve the problem by updating the off-line network embedding methods with the matrix perturbation theory. The DynamicTriad aims at capturing the evolving pattern of the three vertices from open to closed. Both of them incorporate the social homophily and temporal smoothness assumption into the models. The DepthLGP treats the dynamics of network as solving the out-of-sample nodes problems by kernel function and neural network techniques.

We can conclude that network embedding strategies mainly depend on information one aiming at capturing. In essence, network embedding can be seen as feature extraction or dimensionality reduction for network. No method can avoid loss of information during this process. Therefore, preserving useful information such as topology proximity is crucial to network embedding.

For the simplified static homogeneous networks, topology information is that the basic information can reflect networks. Random walk based and matrix factorization based methods are two common techniques in topology information preserving. Based on topology information preserving, other information is incorporated in embedding process such as additional information, edge type information and dynamic information. The more information we crave preserving, the more specific the method is (e.g., HNE and SHINE). However, topology information is the base of all the other embedding methods because topology is the foundation of network.

6. Datasets and evaluations

In this section, several datasets and evaluation tasks are discussed. We first summarize the datasets that are commonly used in existing network embedding algorithms. Then, we discuss the evaluation metrics and downstream tasks in network embedding experimental setups.

6.1. Datasets

Considering different application scenarios, we classify commonly used real-world network datasets into six taxonomies [97]: social networks [98,99], citation networks, collaboration networks [34,100], webpage networks, knowledge graphs and biological networks [101]. It is worth noticing that the same dataset may play different roles in different studies. For example, in LINE, DBLP can be divided into two homogeneous networks: DBLP (Author-Citation) network and DBLP (Paper-Citation) network. It is regarded as a typical heterogeneous network in other network embedding works such as LSHM. However, we only focus on the issues by briefly introducing each dataset without further description. More detailed information can be obtained from the following hyperlink at Table 2.

6.1.1. Social networks

- **BLOGCATALOG** [102]: The nodes of this social network represent bloggers and the edges reflect the relationship among bloggers. The label information concerns the blogger's interests. More details about this dataset can be found at http://networkrepository.com/soc-BlogCatalog.php.
- **TWITTER** [103]: Similar to BLOGCATALOG, Twitter is a social network where node corresponds to the user and the edge corresponds to the following relationship between nodes. More details about this dataset can be found at https://snap.stanford.edu/data/web-flickr.html.
- **FLICKR** [103]: It is another social network dataset in which users contact each other by sharing the photographs. The edge reflects the friendship of nodes representing your off-line friends or strangers who are interested in your photographs. The label of node reflects the users' interest topics. More details about this dataset can be found at https://snap.stanford.edu/data/web-flickr.html.
- **YOUTUBE** [104]: This is a social network constructed by sharing the popular videos among users. Similar to FLICKR, the label of node reflects the preference each user belongs. More details about this dataset can be found at https://snap.stanford.edu/data/com-Youtube.html.

6.1.2. Citation networks

- **DBLP** [104]: DBLP provides the bibliographic information in computer science which is commonly divided into the Paper-Citation subset and the Author-Citation subset. More details about this dataset can be found at https://snap.stanford.edu/data/com-DBLP.html.
- **Cora** [105]: Cora is constructed by machine learning papers. The node is represented by a binary vector with 1433 dimensions indicating the presence/absence of corresponding word and the edge reflects the citation relationships among papers. More details about this dataset can be found at https://linqs-data.soe.ucsc.edu/public/lbc/cora.tgz.
- **CiteSeer** [105]: Similar to Cora, CiteSeer is another computer science dataset where the node is a document represented by a word vector and the edge is the citation relationship of two nodes. More details about this dataset can be found at https://linqs-data.soe.ucsc.edu/public/lbc/citeseer.tgz.

6.1.3. Collaboration networks

- **AMiner** [106]: AMiner is an academic collaboration network where each node is a researcher and the edge reflects the coauthor relationship between nodes. More details about this dataset can be found at https://www.aminer.cn/citation.
- **ArXiv** [107]: ArXiv is also an academic collaboration network which is similar to the AMiner. More details about this dataset can be found at https://www.kaggle.com/Cornell-University/arxiv.

6.1.4. Webpage networks

- **Wikipedia** [108]: Wikipedia is a webpage network which is an epitome or subset of the Internet. Each node is a webpage and the edge is a hyperlink from one page to another. More details about this dataset can be found at http://www.mattmahoney.net/dc/textdata.

Table 2
A summary of datasets and evaluation tasks.

First-layer categorization	Second-layer categorization	Algorithms	Datasets	Evaluation tasks
Network embedding on static homogeneous networks	Network embedding with structural information	DeepWalk	BLOGCATALOG FLICKR YOUTUBE	Node classification
		node2vec	BLOGCATALOG PPI Wikipedia	Node classification
		SDNE	BLOGCATALOG ArXiv 20-NewsGroup	Network reconstruction node classification link prediction network visualization
		DNGR	20-NewsGroup Wine Wikipedia Corpus	Node clustering word similarity network visualization
		HOPE	Synthetic data cora Twitter Tencent Weibo	Network reconstruction link prediction node recommendation
		SiNE	Epinions Slashdot	Link prediction
		LINE	Wikipedia FLICKR YOUTUBE DBLP(AuthorCitation) DBLP(PaperCitation)	Word analogy document classification node classification network visualization
		GraRep	BLOGCATALOG 20-NewsGroup DBLP	Node classification link prediction network visualization
	Network embedding with additional information	TADW	Cora CiteSeer Wikipedia	Node classification
		MMDW	Cora CiteSeer Wikipedia	Node classification network visualization
		LANE	WikipediaBLOGCATALOG	Node classification
Network embedding on static heterogeneous networks	Natural language model based embedding	PTE	20-NewsGroup Wikipedia IMDB RCV1 DBLP MR Twitter	Node classification
		LSHM	DBLP-with no content DBLP-with content Flicker	Node classification
		MetaPath2Vec	Aminer CS DBIS	Node classification node clustering network visualization
		Hin2vec	BLOGCATALOG Yelp DBLP U.S. Patent	Node classification link prediction
	Neural network model based embedding	HNE	BLOGCATALOG NUS-WIDE	Node classification node clustering multimodal search
		SHINE	Weibo Sentiments Wikipedia	Link prediction node recommendation
		Curriculum learning method	DBLP Yelp IMDB	Node classification

(*continued on next page*)

6.1.5. Knowledge graph

- **DBpedia** [109]: DBpedia is a knowledge graph aiming at extracting structure content information based on Wikipedia. The nodes can be various kinds of entities including people, places, films and video games etc. and different types of relationships between these entities. More details about this dataset can be found at https://wiki.dbpedia.org/.

Table 2 (*continued*).

First-layer categorization	Second-layer categorization	Algorithms	Datasets	Evaluation tasks
Network embedding on dynamic networks	None	DANE	BLOGCATALOG Flicker Epinions DBLP	Node classification
	None	DynamicTriad	China Telecom PPDai AMiner	Node Classification Node prediction Link reconstruction Link prediction
	None	DepthLGP	DBLP PPI BLOGCATALOG	Node Classification Link prediction

6.1.6. Biological networks

- **PPI** [110]: This is a sort of biological network where the node is the protein and the edge reflects the physical interactions among them. The label information of node is the hallmark gene sets [111] that corresponds to protein. More details about this dataset can be found at https://snap.stanford.edu/biodata/datasets/10028/10028-PP-Miner.html.

6.2. Evaluations

As mentioned above, we notice that the optimized embedding output can be regarded as feature inputs for the following network analysis tasks. Therefore, a general method of evaluating the proposed network embedding methods is to test the obtained embedding results by applying various downstream tasks (e.g., node classification or link prediction) to compare with other baseline methods.

In this section, we implement Laplacian Eigenmaps, DeepWalk, Node2Vec, HOPE, LINE and TADW in the node classification, node clustering, link prediction and network visualization tasks. These methods are designed for homogeneous networks. As mentioned above, the heterogeneous and dynamic network embedding methods depend largely on the specific network. Therefore, we only choose the homogeneous network embedding methods. For these methods, we used the implementation released by the original authors. The parameters for these methods are tuned to be optimal as much as possible. Note that the dimension of representations of all methods are set to 128 for fair comparison. The dataset we choose to perform these methods is CiteSeer which have been introduced previously. The numbers of vertex set, edge set and attributes set, label set are 3312, 4715, 3703, 6 in CiteSeer. It is worth noting that Laplacian Eigenmaps, DeepWalk, Node2Vec, HOPE and LINE capture the topology information only while TADW preserves topology and attributive information simultaneously.

6.2.1. Network reconstruction

The embedding of a network aims to transform it into a low-dimensional space while preserving information of this original network as much as possible. Therefore, one important evaluation methodology to test the embedding result is to reconstruct a network from its embedding. As we all know, the embedding procedure is essentially extracting the features of network. No matter what features an embedding method aims to obtain (structure, property or other information), the reconstruction of the network can prove it effectively. Furthermore, the original network can be a ground-truth to evaluate the reconstruction results. The commonly used quantitative metrics in network reconstruction are *Precision at k(Pr@k)* [112] and *MeanAveragePrecision (MAP)* [112] which defines as follows:

$$Pr@k = \frac{|E_{pred}(1:k) \cap E_{ext}|}{k} \tag{43}$$

Table 3
Micro-F1 score of node classification on CiteSeer.

Training ratio	10%	20%	30%	40%	50%	60%	70%	80%	90%
Lap	0.429	0.517	0.538	0.542	0.544	0.534	0.543	0.540	0.544
DeepWalk	0.476	0.527	0.564	0.578	0.588	0.594	0.598	0.594	0.604
Node2Vec	0.485	0.542	0.580	0.594	0.612	0.610	0.613	0.622	0.624
HOPE	0.417	0.437	0.442	0.445	0.451	0.455	0.456	0.467	0.468
LINE	0.385	0.438	0.475	0.485	0.503	0.509	0.515	0.519	0.523
TADW	0.654	0.697	0.712	0.723	0.734	0.739	0.740	0.741	0.744

Table 4
Macro-F1 score of node classification on CiteSeer.

Training ratio	10%	20%	30%	40%	50%	60%	70%	80%	90%
Lap	0.376	0.460	0.483	0.489	0.495	0.489	0.502	0.499	0.505
DeepWalk	0.443	0.489	0.522	0.529	0.539	0.543	0.542	0.546	0.551
Node2Vec	0.455	0.505	0.537	0.551	0.566	0.561	0.565	0.572	0.570
HOPE	0.357	0.378	0.386	0.387	0.390	0.393	0.394	0.406	0.403
LINE	0.354	0.406	0.431	0.438	0.452	0.456	0.455	0.457	0.459
TADW	0.610	0.649	0.663	0.677	0.676	0.687	0.686	0.689	0.688

where $E_{pred}(1:k)$ is the top k predictions and E_{ext} is the existed edges.

$$MAP = \frac{\sum_i AP(i)}{|V|} \tag{44}$$

where $AP = \frac{\sum_k Pr@k(i) \cdot \prod E_{pred_i}(k) \in E_{ext_i}}{|k:E_{pred_i} \in E_{ext_i}|}$, E_{pred_i} and E_{ext_i} are predicted edges reconstructed by the embedding results and existed edges for node i respectively. The network reconstruction proximity is a frequently used method to evaluate the network embedding algorithms.

6.2.2. Node classification

The node classification task is the most frequently used evaluation task and an essential application in network analytic research [113]. The node classification consists of two stages: In training stage, a proportional training set of nodes with known labels are trained by selecting appropriate machine learning techniques such as SVM [65] in TADW and Liblear in SDNE to acquire the learning parameters. Then, in testing stage, previous learned classifier associated with corresponding parameters are applied on the testing dataset to verify the classification accuracy. The *micro-F1* and *macro-F1* [114] metrics are frequently used to evaluate the performance of the node classification defined as follows:

$$Macro\text{-}F1 = \frac{\sum_{l \in \mathfrak{L}} F1(l)}{|\mathfrak{L}|} \tag{45}$$

The *macro-F1* is a multi-label classification task evaluation metric, whose value is the average of the sum of all labels' F1 scores. The $F1(l)$ is the F1-score for label l.

$$Micro\text{-}F1 = \frac{2 \times P \times R}{P + R} \tag{46}$$

By computing the number of total true positives, false negatives and false positives, the *micro-F1* obtains a global $F1$ score. Here $P = \frac{\sum_{l\in\mathcal{L}} TP(l)}{\sum_{l\in\mathcal{L}}(TP(l)+FP(l))}$, and $R = \frac{\sum_{l\in\mathcal{L}} TP(l)}{\sum_{l\in\mathcal{L}}(TP(l)+FN(l))}$ are the precision(P) and recall(R) respectively. $TP(l)$, $FP(l)$ and $FN(l)$ correspond to the number of true positives, false positives and false negatives in the instances predicted as label l. According to different features of datasets, node classification can be divided into binary-class classification, multi-class classification and multi-label classification. For example, as shown in Table 2, a large chunk of embedding algorithms such as DeepWalk, node2vec, SDNE and LSHM perform the multi-label classification task on the dataset like the social networks.

The label information can present topics, interests, beliefs or other characteristics of nodes, which is utilized in many application scenarios such as paper recommendation in citation networks [115]. However, only a subset of nodes are labeled in many real-world contexts. Therefore, node classification can be performed on extracting information of unlabeled nodes. The papers (nodes) are classified into one of the following six classes (labels): Agents, AI, DB, IR, ML, HCI in CiteSeer. In this subsection, after obtaining the final representations, we randomly sample 10% to 90% labeled nodes to train a support vector classifier and use rest data to test performance. By comparing this process 10 times, we report the average result of Micro-F1 and Macro-F1 in Tables 3 and 4. TADW captures topology and additional information (attributive information), which actually enhances the embedding performance on node classification task. Furthermore, among methods which only preserve topology information (Lap, DeepWalk, Node2Vec, HOPE and LINE), we can observe that DeepWalk and Node2Vec achieve excellent performance in node classification task.

6.2.3. Node clustering

As an unsupervised task, node clustering aims to cluster the similar nodes according to features in the same group. The ideal result is that nodes in the same groups have a higher cohesion compared with outside nodes [116]. It is especially advantageous when the node labels cannot be easily observed. The general way to implement node clustering task is throwing the embedding result into the classical clustering algorithm such as K-means [117] in DNGR, GraRep and metapath [70]. The most commonly used metrics for evaluating the node clustering performance are the Accuracy (ACC) and normalized mutual information (NMI) [19]. The ACC measures the percentage of correct labels after clustering defined as follows:

$$ACC = \frac{\sum_{i=1}^{n} \delta(r_i, map(l_i))}{n} \tag{47}$$

where l_i and r_i are the obtained label and the ground-truth label respectively. $map(l_i)$ maps each cluster label l_i to corresponding label. $\delta(\cdot,\cdot)$ is a judgment function returning one or zero according to the two parameters whether to be equal. NMI measures the similarity of two clustering results which is defined as follows:

$$NMI = \frac{MI(C, C')}{\max(H(C), H(C'))} \tag{48}$$

where C and C' represent the ground-truth cluster and the clustering result respectively. $H(\cdot)$ is the entropy of C and $MI(C, C')$ is the mutual information metric of C and C'.

As Fig. 6 has shown, in the node clustering, we can find that TADW achieves optimal performance compared with other methods. The DeepWalk and Node2vec achieve similar node clustering results due to similarities of their methods.

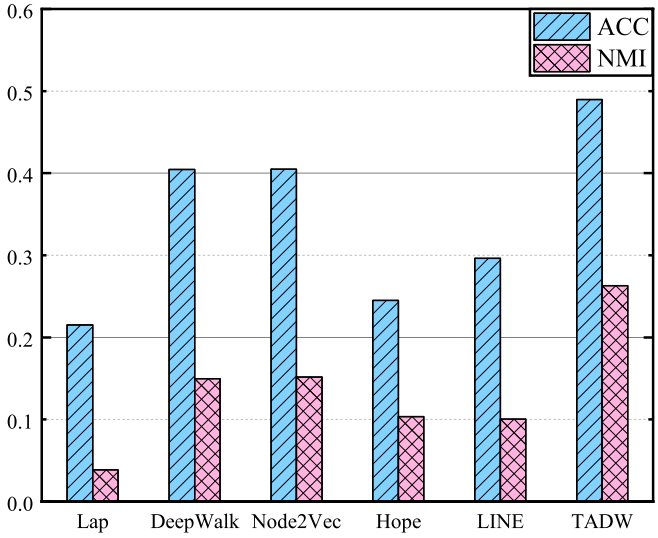

Fig. 6. Node clustering results on CiteSeer.

6.2.4. Link prediction

Similar to the missing node label information mentioned in the node classification, it is not unusual that edges cannot be observed due to various reasons especially in the social networks [118,119]. For example, two friends may have no following relationship in the twitter even though they are familiar with each other. Therefore, based on the similarity of two nodes, the likelihood of the existence of an edge between two nodes can be calculated by analyzing the network structure or property. To some extent, the link prediction can also be regarded as an essential step of network reconstruction [120]. In link prediction task, labeled datasets of edges should be constructed at first. The labeled datasets consists of positive and negative instances. The positive instances are obtained by randomly holding out proportional existing links while the negative instances are sampled from non-existing links. Then, the residual network is used to train the embedding models. The embeddings of nodes are utilized to perform link prediction task in the labeled edge dataset. Based on the cosine similarity function, we can rank both positive and negative instances. To judge the ranking quality, AUC is used to evaluate the rank list as the index of link prediction performance [9].

Furthermore, with the evolution of networks, nodes and edges become uncertain. Therefore, link prediction is a fundamental element in dynamic network analysis [7,91].

We delete 10% existing links in CiteSeer dataset as positive instances and construct an equal number of non-existing links as negative instances. As Fig. 7 has shown, different from node clustering and node classification tasks, we can find that random walk based methods (DeepWalk and Node2Vec) achieve a better result than matrix factorization based methods (TADW). The reason behind it may be that the random walk procedure sample paths based on links among nodes.

6.2.5. Network visualization

Network visualization aims at representing nodes and relations interactively by mapping the network embedding results into two or three dimensional space. After visualization, nodes attached to the same community should be closer than those outside. For example, in SDNE, they represent different clustering nodes as different colors on the 20-NewsGroup with the visualization tool t-SNE. By analyzing the center points and boundary

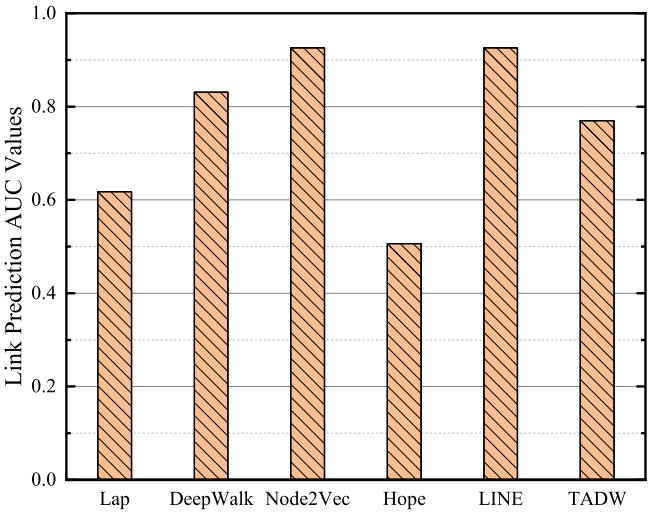

Fig. 7. Link prediction results on CiteSeer.

points in visualization results, SDNE shows the results vividly compared with other methods.

In this subsection, we visualize the embedding vectors of nodes from different methods using t-SNE [121] tookit under the same parameter setting. As we can see in Fig. 8, the visualization result is related to the node clustering results. Lap, DeepWalk, Node2Vec, HOPE and LINE only utilize the topology information to generate embedding results, however, DeepWalk and Node2Vec present a more separable boundaries between different clusters than Lap, HOPE and LINE. TADW achieves the best performance compared with other methods with quite large margin between each cluster. It also proves that additional information considered embedding method is more competitive in network visualization task.

In the evaluation and application section, embedding plays a vital role in extracting the hidden features of the networks. Based on this feature, the similarity of the nodes can be obtained. It means that the relationship can be obtained. Therefore, node classification, node clustering, link prediction and network visualization tasks can be accomplished by using corresponding methods.

7. Future directions

7.1. Granularity

Most of the existing network embedding methods focus on embedding the nodes into low-dimensional space. However, these methods lack the ability to preserve the higher-order structural granularity. Some approaches [122,123] focus on embedding networks based on community structures. The other granularity is based on network motifs. Motif-based methods [124] describe network features using different size of subgraphs, which is a research direction worth examining.

7.2. Task-dependent framework

Due to the complexity of networks mentioned in this paper, designing universal frameworks adapted for various networks becomes even more complicated. Therefore, some network embedding methods design the embedding framework for the specific tasks, such as node classification, node clustering and link prediction. The goal of these task-dependent or task-specific network embedding methods is to extract appropriate and accurate features for predefined tasks.

7.3. Heterogeneity

The heterogeneous network embedding techniques attract high attention in this area due to the universality in heterogeneous network. Even though we introduce some heterogeneous network embedding methods, these methods only capture features available for the textual or the multi-media networks that require high computing cost. Based on these factors, designing heterogeneous network embedding method which can perform effectively on various heterogeneous networks is a promising research direction.

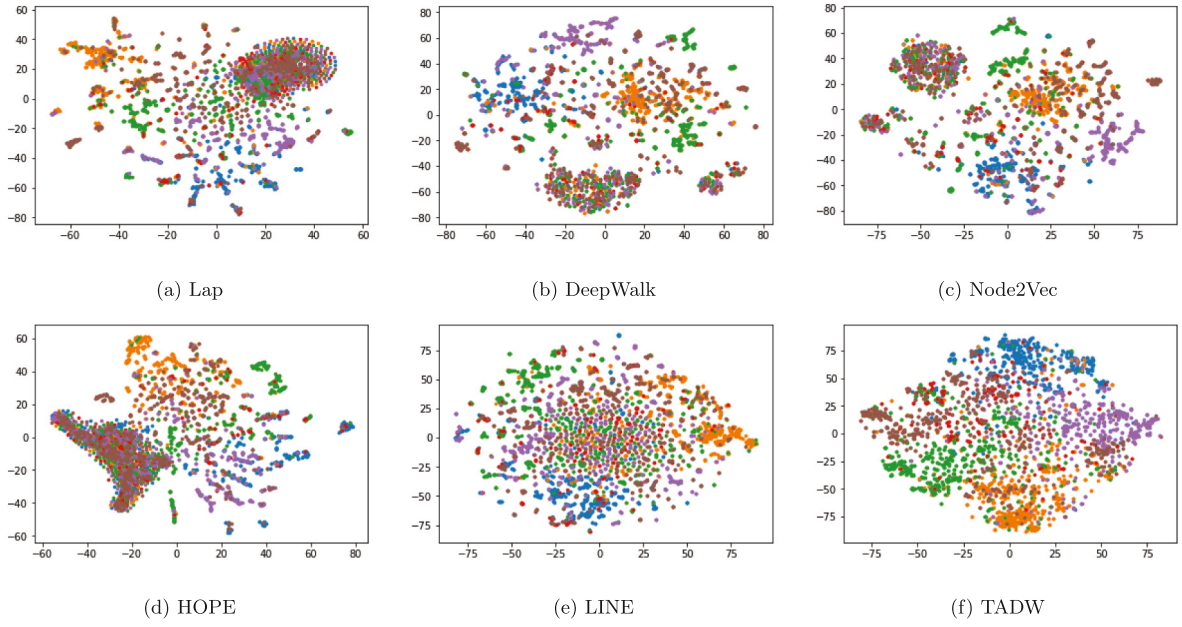

Fig. 8. Visualization of CiteSeer dataset. Each point represents a paper. Different colors correspond to different labels. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

7.4. Dynamics

Dynamics is always a challenge for network embedding tasks considering the complex mechanism with the evolvement of network. Recent dynamic network embedding methods primarily focus on the dynamic properties in the network. Although the existing dynamic network embedding methods devote to reducing the complexity, it is still an inevitable challenge especially when the large-scale network evolves sharply over time. Therefore, it is significant to design dynamic network embedding methods, which can capture more evolutionary features as well as reduce time complexity.

7.5. Scalability

Network contains enormous and multi-source information. This large-scale character enhances the design difficulty of network embedding algorithms and cannot be ignored when we perform network embedding tasks. An effective way to solve this problem is to improve the scalability of the network embedding method. To compute the network distributively, we can deploy large-scale data to multiple servers. Therefore, it is an interesting research direction in the future to design new network embedding methods, in which the nodes and edges are distributed in different shards while maintaining consistency at the same time.

7.6. Uncertainty

Due to the collection error or the privacy protection, the data in the network are noisy and inaccurate. The mining of these uncertain networks has always been an important part of network science [125,126]. However, the uncertain network embedding research is scarce. Recently, researchers start to focus on the uncertain network embedding [127]. Considering the uncertainty in the network, the further embedding research methods for uncertain network embedding are still very faddish and worth investigating.

8. Conclusion

In this survey, we systematically summarize works in network embedding. We provide a clear definition of network embedding and introduce some basic notations and concepts. More importantly, we propose a three-layer categorization framework to organize network embedding methods from the perspective of network evolvement. In the first layer, we classify networks into static networks and dynamic networks. The dynamics of networks reflects the intrinsic features of a network. In the second layer, we classify static networks into two subcategories: homogeneous networks and heterogeneous networks. The heterogeneity of network is another challenge to network embedding due to different types of nodes and edges. In the last layer of the categorization, according to the information that algorithms aim to retain, we classify the static homogeneous network embedding into three groups. After that, we summarize the datasets and evaluation metrics/tasks related to network embedding. Finally, we suggest six promising future research directions in the field of network embedding.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is partially supported by National Natural Science Foundation of China under Grant No. 61872054 and the Fundamental Research Funds for the Central Universities (DUT19LAB23).

References

[1] B. Perozzi, R. Al-Rfou, S. Skiena, DeepWalk: Online learning of social representations, in: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014, pp. 701–710.
[2] M. Ou, P. Cui, J. Pei, Z. Zhang, W. Zhu, Asymmetric transitivity preserving graph embedding, in: ACM SIGKDD International Conference, 2016, pp. 1105–1114.
[3] H.D. Bedru, S. Yu, X. Xiao, D. Zhang, L. Wan, H. Guo, F. Xia, Big networks: A survey, Comp. Sci. Rev. 37 (2020) 100247.
[4] X. Kong, M. Mao, W. Wang, J. Liu, B. Xu, VOPRec: Vector representation learning of papers with text information and structural identity for recommendation, IEEE Trans. Emerg. Top. Comput. PP (99) (2018) 1.
[5] J. Liu, F. Xia, L. Wang, B. Xu, X. Kong, H. Tong, I. King, Shifu2: A network representation learning based model for advisor-advisee relationship mining, IEEE Trans. Knowl. Data Eng. (2019) 1.
[6] W. Wang, J. Liu, F. Xia, I. King, H. Tong, Shifu: Deep learning based advisor-advisee relationship mining in scholarly big data, in: Proceedings of the 26th International Conference on World Wide Web Companion, 2017, pp. 303–310.
[7] J. Ma, P. Cui, W. Zhu, DepthLGP: Learning embeddings of out-of-sample nodes in dynamic networks, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.
[8] S. Bhagat, G. Cormode, S. Muthukrishnan, Node classification in social networks, in: Social Network Data Analytics, Springer, 2011, pp. 115–148.
[9] L. Wang, J. Ren, B. Xu, J. Li, W. Luo, F. Xia, MODEL: Motif-based deep feature learning for link prediction, IEEE Trans. Comput. Soc. Syst. 7 (2) (2020) 503–516.
[10] C.H.Q. Ding, X. He, H. Zha, M. Gu, H.D. Simon, A min-max cut algorithm for graph partitioning and data clustering, in: IEEE International Conference on Data Mining, 2001, pp. 107–114.
[11] X. Kong, F. Xia, Z. Ning, A. Rahim, Y. Cai, Z. Gao, J. Ma, Mobility dataset generation for vehicular social networks based on floating car data, IEEE Trans. Veh. Technol. 67 (5) (2018) 3874–3886.
[12] X. Kong, Y. Shi, S. Yu, J. Liu, F. Xia, Academic social networks: Modeling, analysis, mining and applications, J. Netw. Comput. Appl. 132 (2019) 86–103.
[13] J. Li, H. Dani, X. Hu, J. Tang, Y. Chang, H. Liu, Attributed network embedding for learning in a dynamic environment, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, ACM, 2017, pp. 387–396.
[14] J.B. Tenenbaum, V.D. Silva, J.C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (5500) (2000) 2319–2323.
[15] S.T. Roweis, L.K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326.
[16] M. Belkin, P. Niyogi, Laplacian eigenmaps and spectral techniques for embedding and clustering, Adv. Neural Inf. Process. Syst. 14 (6) (2001) 585–591.
[17] D. Haussler, Convolution kernels on discrete structures, Tech. Rep. 7 (1999) 95–114.
[18] J. Liu, X. Kong, X. Feng, X. Bai, W. Lei, Q. Qing, I. Lee, Artificial intelligence in the 21st century, IEEE Access 6 (99) (2018) 34403–34421.
[19] C. Peng, W. Xiao, P. Jian, W. Zhu, A survey on network embedding, IEEE Trans. Knowl. Data Eng. PP (99) (2017) 1.
[20] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, Q. Mei, LINE: Large-scale information network embedding, in: International Conference on World Wide Web, 2015.
[21] C. Yang, D. Zhao, D. Zhao, E.Y. Chang, E.Y. Chang, Network representation learning with rich text information, in: International Conference on Artificial Intelligence, 2015, pp. 2111–2117.
[22] X. Huang, J. Li, X. Hu, Label informed attributed network embedding, in: Tenth ACM International Conference on Web Search and Data Mining, 2017, pp. 731–739.
[23] S. Yan, D. Xu, B. Zhang, H. Zhang, Q. Yang, S. Lin, Graph embedding and extensions: A general framework for dimensionality reduction, IEEE Trans. Pattern Anal. Mach. Intell. 29 (1) (2007) 40–51.
[24] F. Xia, L. Liu, J. Li, A.M. Ahmed, L.T. Yang, J. Ma, BEEINFO: Interest-based forwarding using artificial bee colony for socially aware networking, IEEE Trans. Veh. Technol. 64 (3) (2015) 1188–1200.
[25] X. Kong, J. Zhang, D. Zhang, Y. Bu, Y. Ding, F. Xia, The gene of scientific success, ACM Trans. Knowl. Discov. Data (TKDD) 14 (4) (2020) 19.

[26] H. Cai, V.W. Zheng, K.C.-C. Chang, A comprehensive survey of graph embedding: Problems, techniques, and applications, IEEE Trans. Knowl. Data Eng. 30 (9) (2018) 1616–1637.
[27] P. Goyal, E. Ferrara, Graph embedding techniques, applications, and performance: A survey, Knowl.-Based Syst. 151 (2018) 78–94.
[28] F. Xia, J. Liu, H. Nie, Y. Fu, L. Wan, X. Kong, Random walks: A review of algorithms and applications, IEEE Trans. Emerg. Top. Comput. Intell. 4 (2) (2019) 95–107.
[29] F. Xia, Z. Chen, W. Wang, J. Li, L.T. Yang, MVCWalker: Random walk-based most valuable collaborators recommendation exploiting academic factors, IEEE Trans. Emerg. Top. Comput. 2 (3) (2014) 364–375.
[30] Q. Wang, Z. Mao, B. Wang, L. Guo, Knowledge graph embedding: A survey of approaches and applications, IEEE Trans. Knowl. Data Eng. 29 (12) (2017) 2724–2743.
[31] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, in: Advances in Neural Information Processing Systems, 2013, pp. 2787–2795.
[32] S. Cao, W. Lu, Q. Xu, GraRep: Learning graph representations with global structural information, in: ACM International on Conference on Information and Knowledge Management, 2015, pp. 891–900.
[33] J. Xu, S. Yu, K. Sun, J. Ren, I. Lee, S. Pan, F. Xia, Multivariate relations aggregation learning in social networks, in: Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020, 2020, pp. 77–86.
[34] W. Wang, S. Yu, T.M. Bekele, X. Kong, F. Xia, Scientific collaboration patterns vary with scholars'academic ages, Scientometrics 112 (1) (2017) 329–343.
[35] L. Wan, Y. Yuan, F. Xia, H. Liu, To your surprise: Identifying serendipitous collaborators, IEEE Trans. Big Data (2019) 1.
[36] S. Yu, F. Xia, K. Zhang, Z. Ning, J. Zhong, C. Liu, Team recognition in big scholarly Data: Exploring collaboration intensity, in: 2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress, DASC/PiCom/DataCom/CyberSciTech, 2017, pp. 925–932.
[37] T. Lin, H. Zha, Riemannian manifold learning, IEEE Trans. Pattern Anal. Mach. Intell. 30 (5) (2008) 796–809.
[38] M.A.A. Cox, T.F. Cox, Multidimensional scaling, J. R. Stat. Soc. 46 (2) (2001) 1050–1057.
[39] A. Smola, Kernels and regularization on graphs, Lecture Notes in Comput. Sci. 2777 (2003) 144–158.
[40] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, 2013, arXiv:1301.3781.
[41] D.A. Huffman, A method for the construction of minimum-redundancy codes, Resonance 11 (2) (2006) 91–99.
[42] A. Grover, 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.
[43] W.L. Hamilton, R. Ying, J. Leskovec, Representation learning on graphs: Methods and applications, 2017, arXiv:1709.05584.
[44] Y. Bengio, Learning Deep Architectures for AI, Now Publishers Inc, 2009.
[45] D. Wang, P. Cui, W. Zhu, Structural deep network embedding, in: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 1225–1234.
[46] C. Davis, The norm of the Schur product operation, Numer. Math. 4 (1) (1962) 343–344.
[47] L.Y. Deng, The cross-entropy method: A unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning, Technometrics 48 (1) (2004) 147–148.
[48] C. Eckart, G. Young, The approximation of one matrix by another of lower rank, Psychometrika 1 (3) (1936) 211–218.
[49] S. Cao, W. Lu, Q. Xu, Deep neural networks for learning graph representations, in: Thirtieth AAAI Conference on Artificial Intelligence, 2016, pp. 1145–1152.
[50] L. Page, The pagerank citation ranking: Bringing order to the web, in: Stanford Digital Libraries Working Paper, Vol. 9, No. 1, 1998, pp. 1–14.
[51] P. Hanks, P. Hanks, Word association norms, mutual information, and lexicography, in: Meeting on Association for Computational Linguistics, 1990, pp. 76–83.
[52] L. Katz, A new status index derived from sociometric analysis, Psychometrika 18 (1) (1953) 39–43.
[53] D. Liben-Nowell, J. Kleinberg, The Link-Prediction Problem for Social Networks, John Wiley and Sons, Inc., 2007, pp. 1019–1031.
[54] L. A.A.damic, E. Adar, Friends and neighbors on the web, Social Networks 25 (3) (2003) 211–230.
[55] J. Leskovec, D. Huttenlocher, J. Kleinberg, Predicting positive and negative links in online social networks, in: International Conference on World Wide Web, 2010, pp. 641–650.
[56] H. Ma, M.R. Lyu, I. King, Learning to recommend with trust and distrust relationships, in: ACM Conference on Recommender Systems, 2009, pp. 189–196.
[57] S. Wang, J. Tang, C. Aggarwal, Y. Chang, H. Liu, Signed network embedding in social media, in: Proceedings of the 2017 SIAM International Conference on Data Mining, SIAM, 2017, pp. 327–335.
[58] HEIDERF, Attitudes and cognitive organization, J. Psychol. 21 (1) (1977) 3–8.
[59] D. Cartwright, F. Harary, Structural balance: a generalization of Heider's theory, Psychol. Rev. 63 (5) (1977) 9–25.
[60] S. Kullback, R.A. Leibler, On information and sufficiency, Ann. Math. Stat. 22 (1) (1951) 79–86.
[61] J. Liu, X. Kong, F. Xia, X. Bai, L. Wang, Q. Qing, I. Lee, Artificial intelligence in the 21st century, IEEE Access 6 (2018) 34403–34421.
[62] H.F. Yu, P. Jain, P. Kar, I.S. Dhillon, Large-scale multi-label learning with missing labels, 2013, pp. 593–601.
[63] N. Natarajan, I.S. Dhillon, Inductive matrix completion for predicting gene–disease associations, Bioinformatics 30 (12) (2014) i60–i68.
[64] C. Tu, W. Zhang, Z. Liu, M. Sun, Max-margin deepwalk: Discriminative learning of network representation, in: International Joint Conference on Artificial Intelligence, 2016, pp. 3889–3895.
[65] M.A. Hearst, S.T. Dumais, E. Osuna, J. Platt, B. Scholkopf, Support vector machines, IEEE Intell. Syst. Appl. 13 (4) (1998) 18–28.
[66] J. Liu, J. Tian, X. Kong, I. Lee, F. Xia, Two decades of information systems: a bibliometric review, Scientometrics 118 (2) (2019) 617–643.
[67] J. Tang, M. Qu, Q. Mei, PTE: predictive text embedding through large-scale heterogeneous text networks, in: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2015 pp. 1165–1174.
[68] Y. Jacob, L. Denoyer, P. Gallinari, Learning latent representations of nodes for classifying in heterogeneous social networks, in: Proceedings of the 7th ACM International Conference on Web Search and Data Mining, ACM, 2014, pp. 373–382.
[69] M.A. Vorontsov, V.P. Sivokon, Stochastic parallel-gradient-descent technique for high-resolution wave-front phase-distortion correction, J. Opt. Soc. Amer. A 15 (10) (1998) 2745–2758.
[70] Y. Sun, J. Han, Mining heterogeneous information networks: Principles and methodologies, Acm Sigkdd Explor. Newslett. 14 (2) (2012) 20–28.
[71] A. Swami, A. Swami, A. Swami, metapath2vec: Scalable representation learning for heterogeneous networks, in: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017, pp. 135–144.
[72] T.-y. Fu, W.-C. Lee, Z. Lei, Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, 2017, pp. 1797–1806.
[73] S. Chang, W. Han, J. Tang, G.J. Qi, C.C. Aggarwal, T.S. Huang, Heterogeneous network embedding via deep architectures, in: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2015, pp. 119–128.
[74] B. Schölkopf, J. Platt, T. Hofmann, Greedy layer-wise training of deep networks, in: International Conference on Neural Information Processing Systems, 2006, pp. 153–160.
[75] G. Hinton, R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.
[76] X.G.A. Bordes, Y. Bengio, Deep Sparse Rectifier Networks, AISTATS, 2011.
[77] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, in: International Conference on Neural Information Processing Systems, 2012, pp. 1097–1105.
[78] H. Wang, F. Zhang, M. Hou, X. Xie, M. Guo, Q. Liu, Shine: Signed heterogeneous information network embedding for sentiment link prediction, in: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, ACM, 2018, pp. 592–600.
[79] Z. Wang, Y. Zhang, H. Chen, Z. Li, F. Xia, Deep user modeling for content-based event recommendation in event-based social networks, in: IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, 2018, pp. 1304–1312.
[80] F. Zhang, N.J. Yuan, D. Lian, X. Xie, W.Y. Ma, Collaborative knowledge base embedding for recommender systems, in: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 353–362.
[81] P. Wang, J. Guo, Y. Lan, J. Xu, S. Wan, X. Cheng, Learning hierarchical representation model for nextbasket recommendation, in: International ACM SIGIR Conference on Research and Development in Information Retrieval, 2015, pp. 403–412.
[82] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529.
[83] F. Wu, J. Song, Y. Yang, X. Li, Z. Zhang, Y. Zhuang, Structured embedding via pairwise relations and long-range interactions in knowledge base, in: Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015, pp. 1663–1670.
[84] J. Liu, J. Ren, W. Zheng, L. Chi, I. Lee, F. Xia, Web of scholars: A scholar knowledge graph, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 2153–2156.
[85] S. Gelly, D. Silver, Combining online and offline knowledge in UCT, in: International Conference on Machine Learning, 2007, pp. 273–280.

[86] R. Sutton, A. Barto, Reinforcement learning: An introduction, bradford book, Mach. Learn. 16 (1) (2005) 285–286.
[87] S. Tang, B. Andres, M. Andriluka, B. Schiele, Multi-person tracking by multicut and deep matching, in: European Conference on Computer Vision, 2016, pp. 100–111.
[88] F. Xia, A. Rahim, X. Kong, M. Wang, Y. Cai, J. Wang, Modeling and analysis of large-scale urban mobility for green transportation, IEEE Trans. Ind. Inf. 14 (4) (2018) 1469–1481.
[89] F. Xia, J. Wang, X. Kong, D. Zhang, Z. Wang, Ranking station importance with human mobility patterns using subway network datasets, IEEE Trans. Intell. Transp. Syst. 21 (7) (2020) 2840–2852.
[90] W. Wang, J. Liu, T. Tang, S. Tuarob, F. Xia, Z. Gong, I. King, Attributed collaboration network embedding for academic relationship mining, ACM Trans. Web 1 (1) (2020).
[91] L. Zhou, Y. Yang, X. Ren, F. Wu, Y. Zhuang, Dynamic network embedding by modeling triadic closure process, in: Thirty-Second AAAI Conference on Artificial Intelligence.
[92] S. Yu, J. Xu, C. Zhang, F. Xia, Z. Almakhadmeh, A. Tolba, Motifs in big networks: Methods and applications, IEEE Access 7 (2019) 183322–183338.
[93] F. Xia, H. Wei, S. Yu, D. Zhang, B. Xu, A survey of measures for network motifs, IEEE Access 7 (2019) 106576–106587.
[94] B. Schölkopf, J. Platt, T. Hofmann, Relational learning with Gaussian processes, in: Conference on Advances in Neural Information Processing Systems, 2005, pp. 137–144.
[95] K. Yu, W. Chu, Gaussian process models for link analysis and transfer learning, in: International Conference on Neural Information Processing Systems, 2007, pp. 1657–1664.
[96] M. Seeger, Gaussian processes for machine learning, Publ. Am. Stat. Assoc. 103 (481) (2008) 429.
[97] F. Xia, W. Wang, T.M. Bekele, H. Liu, Big scholarly data: A survey, IEEE Trans. Big Data 3 (1) (2017) 18–35.
[98] F. Xia, N.Y. Asabere, H. Liu, Z. Chen, W. Wang, Socially aware conference participant recommendation with personality traits, IEEE Syst. J. 11 (4) (2014) 2255–2266.
[99] D. Zhang, T. Guo, H. Pan, J. Hou, Z. Feng, L. Yang, H. Lin, F. Xia, Judging a book by its cover: The effect of facial perception on centrality in social networks, in: The World Wide Web Conference, 2019, pp. 2290–2300.
[100] S. Yu, H.D. Bedru, I. Lee, F. Xia, Science of scientific team science: A survey, Comp. Sci. Rev. 31 (2019) 72–83.
[101] B. Xu, K. Li, W. Zheng, X. Liu, Y. Zhang, Z. Zhao, Z. He, Protein complexes identification based on go attributed network embedding, BMC Bioinform. 19 (1) (2018) 535.
[102] L. Tang, H. Liu, Relational learning via latent social dimensions, in: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28–July, 2009 pp. 817–826.
[103] J. McAuley, J. Leskovec, Image labeling on a network: using social-network metadata for image classification, in: European Conference on Computer Vision, Springer, 2012, pp. 828–841.
[104] J. Yang, J. Leskovec, Defining and evaluating network communities based on ground-truth, Knowl. Inf. Syst. 42 (1) (2015) 181–213.
[105] A.K. Mccallum, K. Nigam, J. Rennie, K. Seymore, Automating the construction of internet portals with machine learning, Inf. Retr. 3 (2) (2000) 127–163.
[106] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, Z. Su, ArnetMiner: extraction and mining of academic social networks, in: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2008, pp. 990–998.
[107] J. Leskovec, J. Kleinberg, C. Faloutsos, Graph evolution: Densification and shrinking diameters, ACM Trans. Knowl. Discov. Data (TKDD) 1 (1) (2007) 2.
[108] D. Milne, I.H. Witten, Learning to link with wikipedia, in: Proceedings of the 17th ACM Conference on Information and Knowledge Management, 2008, pp. 509–518.
[109] J. Lehmann, DBpedia - A large-scale, multilingual knowledge base extracted from wikipedia, Seman. Web 6 (2) (2015) 167–195.
[110] D. Szklarczyk, J.H. Morris, H. Cook, M. Kuhn, S. Wyder, M. Simonovic, A. Santos, N.T. Doncheva, A. Roth, P. Bork, L.J. Jensen, C. vonMering, The STRING database in 2017: quality-controlled protein–protein association networks, made broadly accessible, Nucleic Acids Res. 45 (D1) (2016) D362–D368.
[111] A. Liberzon, A. Subramanian, R. Pinchback, H. Thorvaldsdóttir, P. Tamayo, J.P. Mesirov, Molecular signatures database (MSigDB) 3.0, Bioinformatics 27 (12) (2011) 1739.
[112] A. Turpin, F. Scholer, User performance versus precision measures for simple search tasks, in: International ACM SIGIR Conference on Research and Development in Information Retrieval, 2006, pp. 11–18.
[113] J. Macqueen, Some methods for classification and analysis of multivariate observations, in: Proc. of Berkeley Symposium on Mathematical Statistics and Probability, 1966, pp. 281–297.
[114] J. Han, J. Pei, M. Kamber, Data Mining: Concepts and Techniques, Elsevier, 2011.
[115] F. Xia, H. Liu, I. Lee, L. Cao, Scientific article recommendation: Exploiting common author relations and historical preferences, IEEE Trans. Big Data 2 (2) (2016) 101–112.
[116] S. Yu, F. Xia, H. Liu, Academic team formulation based on Liebig's barrel: Discovery of anticask effect, IEEE Trans. Comput. Soc. Syst. 6 (5) (2019) 1083–1094.
[117] J.A. Hartigan, M.A. Wong, Algorithm AS 136: A K-means clustering algorithm, J. R. Stat. Soc. 28 (1) (1979) 100–108.
[118] F. Xia, L. Liu, B. Jedari, S.K. Das, PIS: A multi-dimensional routing protocol for socially-aware networking, IEEE Trans. Mob. Comput. 15 (11) (2016) 2825–2836.
[119] F. Xia, H.B. Liaqat, J. Deng, J. Wan, S.K. Das, Overhead control with reliable transmission of popular packets in ad-hoc social networks, IEEE Trans. Veh. Technol. 65 (9) (2016) 7647–7661.
[120] B. Xu, L. Li, J. Liu, L. Wan, X. Kong, F. Xia, Disappearing link prediction in scientific collaboration networks, IEEE Access 6 (2018) 69702–69712.
[121] L.v.d. Maaten, G. Hinton, Visualizing data using t-SNE, J. Mach. Learn. Res. 9 (Nov) (2008) 2579–2605.
[122] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, S. Yang, Community preserving network embedding, in: The AAAI Conference on Artificial Intelligence, 2017.
[123] F. Xia, A.M. Ahmed, L.T. Yang, Z. Luo, Community-based event dissemination with optimal load balancing, IEEE Trans. Comput. 64 (7) (2015) 1857–1869.
[124] A. Paranjape, A.R. Benson, J. Leskovec, Motifs in temporal networks, in: Tenth ACM International Conference on Web Search and Data Mining, 2017, pp. 601–610.
[125] P. Boldi, F. Bonchi, A. Gionis, T. Tassa, Injecting uncertainty in graphs for identity obfuscation, Proc. VLDB Endow. 5 (11) (2012) 1376–1387.
[126] F. Xia, B. Jedari, L.T. Yang, J. Ma, R. Huang, A signaling game for uncertain data delivery in selfish mobile social networks, IEEE Trans. Comput. Soc. Syst. 3 (2) (2016) 100–112.
[127] J. Hu, R. Cheng, Z. Huang, Y. Fang, S. Luo, On embedding uncertain graphs, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, 2017, pp. 157–166.