Network Representation Learning: A Survey

Journal:	*Transactions on Big Data*
Manuscript ID	TBD-2017-12-0310
Manuscript Types:	Survey-Tutorial Paper
Keywords:	Information networks, Network representation learning, Network embedding

Network Representation Learning: A Survey

Daokun Zhang, Jie Yin, Xingquan Zhu *Senior Member, IEEE*, Chengqi Zhang *Senior Member, IEEE*

Abstract—With the widespread use of information technologies, information networks have increasingly become popular to capture complex relationships across various disciplines, such as social networks, citation networks, telecommunication networks, and biological networks. Analyzing these networks sheds light on different aspects of social life such as the structure of society, information diffusion, and different patterns of communication. However, the large scale of information networks often makes network analytic tasks computationally expensive and intractable. Recently, network representation learning has been proposed as a new learning paradigm that embeds network vertices into a low-dimensional vector space, by preserving network topology structure, vertex content, and other side information. This facilitate the original network to be easily handled in the new vector space for further analysis. In this survey, we perform a thorough review of the current literature on network representation learning in the field of data mining and machine learning. We propose a new categorization to analyze and summarize state-of-the-art network representation learning techniques according to the methodology they employ and the network information they preserve. Finally, to facilitate research on this topic, we summarize benchmark datasets and evaluation methodologies, and discuss open issues and future research directions in this field.

Index Terms—Information networks, network representation learning, network embedding.

1 Introduction

Nowadays, information networks are becoming ubiquitous in a large spectrum of real-world applications in forms of social networks, citation networks, telecommunication networks and biological networks, etc. The scale of these information networks ranges from hundreds of vertices to millions or even billions of vertices [1]. Analyzing information networks plays a crucial role in a variety of emerging applications across many disciplines. For example, in social networks, classifying users into meaningful social groups is useful for many important tasks, such as user search, targeted advertising and recommendations; in communication networks, detecting community structures can help better understand the rumor spreading process; in biological networks, inferring interactions between proteins can facilitate new treatments for diseases. Nevertheless, efficient analysis of these networks heavily relies on the ways how networks are represented. Often, a discrete adjacency matrix is used to represent a network, which only captures neighboring relationships between vertices. However, this simple representation cannot embody more complex, higher-order structure relationships, such as paths, frequent substructure etc. As a result, such a traditional routine often makes many network analytic tasks computationally expensive and intractable over large-scale networks. Taking the community detection task as an example, most of the existing algorithms involve calculating the spectral decomposition of a matrix [2] with at least quadratic time complexity with respect to the number of vertices. This incurs too much computational overhead and cannot scale well to large-scale networks with millions of vertices.

Recently, network representation learning (NRL) has aroused a lot of research interest, which aims to learn latent, low-dimensional representations of network vertices, while preserving network topology structure, vertex content, and other side information. After new vertex representations are learned, network analytic tasks can be easily and efficiently carried out by applying conventional vector-based machine learning algorithm in the new representation space. This obviates the necessity for deriving complex algorithms that are applied directly on the original network.

Earlier work related to network representation learning dates back to the early 2000s, when researchers proposed graph embedding algorithms as part of dimensionality reduction techniques. Given a set of i.i.d. (independent and identically distributed) data points as input, graph embedding algorithms first calculate the similarity between pairwise data points to construct an affinity graph, e.g., the k-nearest neighbor graph, and then embed the affinity graph into a new space having much lower dimensionality. The idea is to find a low-dimensional manifold structure hidden in the high-dimensional data geometry reflected by the constructed graph, so that connected nodes are kept closer to each other in the new embedding space. Isomap [3], Locally Linear Embedding (LLE) [4] and Laplacian Eigenmap [5] are examples of algorithms based on this rationale. However, graph embedding algorithms are designed on i.i.d. data mainly for dimensionality reduction purpose. Most of these algorithm usually have at least quadratic time complexity with respect to the number of vertices, so the scalability is a major issue when they are applied to large-scale networks.

Since 2008, significant research efforts have shifted to the development of effective and scalable representation learning techniques that are directly designed for complex information networks. Many network representation learning algorithms, e.g., [6], [7], [8], [9], have been proposed to embed existing networks, showing promising performance for various applications. These algorithms embed a network

- *Daokun Zhang and Chengqi Zhang are with Centre for Artificial Intelligence, FEIT, University of Technology Sydney, Australia*
  Email: Daokun.Zhang@student.uts.edu.au, Chengqi.Zhang@uts.edu.au.
- *Jie Yin is with Data61, CSIRO, Australia.*
  Email: Jie.Yin@csiro.au.
- *Xingquan Zhu is with Department of CEECS, Florida Atlantic University, USA and School of Computer Science, Fudan University, China.*
  Email: xqzhu@cse.fau.edu.

Manuscript received December 4, 2017.

into a latent, low-dimensional space that preserves structure proximity and attribute affinity, such that the original vertices of the network can be represented as low-dimensional vectors. The resulting compact, low-dimensional vector representations can be then taken as features to any vector-based machine learning algorithms. This paves the way for a wide range of network analytic tasks to be easily and efficiently tackled in the new vector space, such as node classification [10], [11], link prediction [12], [13], clustering [2], visualization [14]. Using vector representation to represent complex networks has now been gradually advanced to many other domains, such as point-of-interest recommendation in urban computing [15], and knowledge graph search [16] in knowledge engineering and database systems.

1.1 Challenges

Despite its great potential, network representation learning is inherently difficult and is confronted with several key challenges.

(i) **Structure-preserving**: To learn informative vertex representations, network representation learning should preserve network structure similarity, which means that nodes similar/close to each other in the original structure space should also be represented similarly in the learned vector space. However, as stated in [17], [18], the structure-level similarity between vertices is reflected not only at the local neighborhood structure but also at the more global community structure. Therefore, the local and global structure should be simultaneously preserved in network representation learning.

(ii) **Content-preserving**: Besides structure information, vertices and edges of many information networks are attached with rich content on attributes. Vertex attributes not only exert huge impacts on the forming of networks, but also provide direct evidence to measure attribute-level similarity between vertices. Therefore, if properly imported, attribute content can compensate network structure to render more informative vertex representations. However, due to heterogeneity of the two information sources, how to effectively leverage vertex attributes and make them compensate rather than deteriorate network structure is an open research problem.

(iii) **Data sparsity**: For many real-world information networks, due to the privacy or legal restrictions, the problem of data sparsity exists in both network structure and vertex content. At the structure level, sometimes only very limited links are observed, which makes it hard to discover the structure-level relatedness between vertices that are not explicitly connected. At the vertex content level, many values of vertex attributes are usually missing, which increases the difficulty of measuring content-level vertex similarity. Thus, it is challenging for network representation learning to overcome the data sparsity problem.

(iv) **Scalability**: Many information networks, especially social networks, consist of millions or billions of vertices. The large scale of information networks challenges not only the traditional network analytic tasks but also the newborn network representation learning task. Without special concern, learning vertex representations for large-scale information networks with limited computing resources may cost months of time, which is definitely unacceptable, especially for the case involving a large number of trails for tuning parameters. Therefore, it is necessary to design network representation learning algorithms that can learn vertex representations efficiently and meanwhile guarantee the effectiveness for large-scale information networks.

1.2 Our Contribution

In this survey, we give a comprehensive review of the state-of-the-art network representation learning techniques, with a focus on the learning of vertex representations. This survey covers not only early work on preserving network structure, but also a new surge of recent studies that leverage side information such as vertex content and labels. By doing so, we hope to provide a useful guideline for the data mining and machine learning community to understand (1) the taxonomy of network representation learning methods, (2) the characteristics, uniqueness, and the niche of different types of network embedding methods, and (3) the resources and future challenges to stimulate research in the area. In particular, this survey has three main contributions:

1) We propose a new categorization of the existing network representation learning techniques according to the methodology that they employ and the network information that they preserve;
2) We provide a detailed and thorough study of the state-of-the-art network representation learning algorithms including their computational models and complexity, as well as compare their differences.
3) To foster future research on this topic, we summarize published benchmark datasets and evaluation methods that have been used for the validation of network representation learning techniques, along with a discussion of applications and potential research directions.

1.3 Related Surveys and Differences

There are a few related surveys about graph embedding and representation learning techniques in the very recent literature. The first is the work by [19], which reviews a few representative methods for network representation learning. This survey does not intend to provide a thorough summary of the various representation learning models and algorithms in the literature, but aims to visit some key concepts around the idea of representation learning and its connections to other related field such as dimensionality reduction, deep learning, and network science. Another two related surveys [20], [21] mainly review the graph embedding techniques aiming to preserve network structure, but do not cover a new surge of recent research work that exploits other side information, such as vertex attributes and/or vertex labels, to harness the learning of network representations.

Unlike previous surveys, we aim to perform a thorough review of the state-of-the-art literature on network representation learning, including not only early work on preserving network structure, but also emerging studies that leverage attribute affinity and label proximity to seek more effective network representations. We provide a systematic categorization and detailed analysis of the existing techniques from two algorithmic perspectives: the information sources and the methodology. We also summarize dataset resources and evaluation methodologies, as well as point out potential research directions for future work. We hope this survey can lay a good foundation to foster future research on network representation learning in the research community.

1.4 Organization of the Survey

The rest of this survey is organized as follows. In Section 2, we provide preliminaries and definitions required to understand the problem and the models discussed next. Section 3 proposes a taxonomy to categorize the existing network representation learning techniques. Section 4 and Section 5 reviews representative algorithms in two categories, respectively. We describe a list of successful applications of network representation learning in Section 6. The benchmark datasets and evaluation methods that researchers have used to validate network representation learning are provided in Section 7. Finally, we conclude the survey and discuss potential research directions in Section 8.

2 Notations and Definitions

In this section, as preliminaries, we first give the definitions of important terminologies that are used to discuss the models next, followed by a formal definition of the network representation learning problem. For ease of presentation, we first define a list of common notations that will be used throughout the survey, as shown in Table 1.

TABLE 1
A summary of common notations

G	The given information network
V	Set of vertices in the given information network
E	Set of edges in the given information network
$\|V\|$	Number of vertices
$\|E\|$	Number of edges
m	Number of vertex attributes
d	Dimension of learned vertex representations
$X \in \mathbb{R}^{\|V\|\times m}$	The vertex attribute matrix
$\mathcal{Y}$	Set of vertex labels
$\|\mathcal{Y}\|$	Number of vertex labels
$Y \in \mathbb{R}^{\|V\|\times\|\mathcal{Y}\|}$	The vertex label matrix

Definition 1 (Information Network). An information network is defined as $G = (V, E, X, Y)$, where V denotes a set of vertices, and $|V|$ denotes the number of vertices in network G. $E \subseteq (V \times V)$ denotes a set of edges connecting the vertices. $X \in \mathbb{R}^{|V|\times m}$ is the vertex attribute matrix, where m is the number of attributes, and the element X_{ij} is the value of the i-th vertex on the j-th attribute. $Y \in \mathbb{R}^{|V|\times|\mathcal{Y}|}$ is the vertex label matrix with $\mathcal{Y}$ being a set of labels. If the i-th vertex has the k-th label, the element $Y_{ik} = 1$; otherwise, $Y_{ik} = -1$. Due to privacy concern or information access difficulty, vertex attribute matrix X is often sparse and vertex label matrix Y is usually unobserved or partially observed. For each $(v_i, v_j) \in E$, if information network G is undirected, we have $(v_j, v_i) \in E$; if G is directed, (v_j, v_i) unnecessarily belongs to E. Each edge $(v_i, v_j) \in E$ is also associated with a weight w_{ij}, which is equal to 1, if the information network is binary (unweighted).

Intuitively, the generation of information networks is not groundless but is guided or dominated by certain latent mechanisms. Although the latent mechanisms are hardly known perfectly, they can be reflected by some network properties that widely exist in information networks. Hence, the common network properties are essential for the learning of vertex representations that are informative enough to accurately interpret the forming of information networks. Below, we introduce several common network properties.

Definition 2 (First-order Proximity). The first-order proximity is the local pairwise proximity between two connected vertices [1]. For each vertex pair (v_i, v_j), if $(v_i, v_j) \in E$, the first-order proximity between v_i and v_j is w_{ij}; otherwise, the first-order proximity between v_i and v_j is 0. The first-order proximity captures the direct neighboring relationships between vertices.

Definition 3 (Second-order Proximity and High-order Proximity). The second-order proximity captures the 2-step relations between each pair of vertices [1]. For each vertex pair (v_i, v_j), the second order proximity is determined by the number of common neighbors shared by the two vertices, which can also be measured by the 2-step transition probability from v_i to v_j equivalently. Compared with the second-order proximity, the high-order proximity [22] captures more global structure, which explores k-step $(k \geq 3)$ relations between each pair of vertices. For each vertex pair (v_i, v_j), the higher-order proximity is measured by the k-step $(k \geq 3)$ transition probability from vertex v_i to vertex v_j, which can also be reflected by the number of k-step $(k \geq 3)$ paths from v_i to v_j. The second-order proximity and the high-order proximity capture the similarity between a pair of, indirectly connected, vertices with similar structural roles.

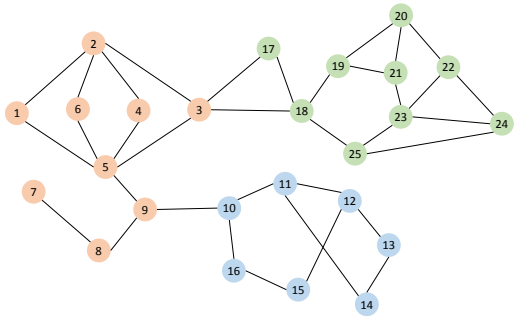
Definition 4 (Intra-community Proximity). The intra-community proximity is the pairwise proximity between two vertices in a same community. Many networks have community structure, where vertex-vertex connections within the same community are dense, but connections to vertices outside the community are sparse [23]. As cluster structure, a community preserves certain kinds of common properties of vertices within it. For example, in social networks, communities might represent social groups by interest or background; in citation networks, communities might represent related papers on a same topic. The intra-community proximity captures such cluster structure by preserving the common property shared by vertices within a same community [24].

In addition to network structure, vertex attributes can provide direct evidence to measure content-level similarity between vertices. As shown in [7], [18], [25], vertex attributes and network structure can help each other filter out noisy information and compensate each other to jointly learn informative vertex representations. Different from the hidden community structure, vertex labels provide direct information about the semantic categorization of each network vertex to certain classes or groups. Vertex labels are strongly influenced by and inherently correlated to both network structure and vertex attributes [26]. Though vertex labels are usually partially observed, when coupled with network structure and vertex attributes, they encourage a network structure and vertex attribute consistent labeling, and help learn informative and discriminative vertex representations.

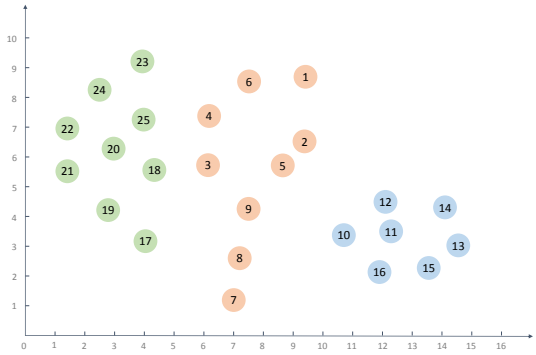
Definition 5 (Network Representation Learning (NRL)). Given an information network $G = (V, E, X, Y)$, by integrating network structure in E, vertex attributes in X and vertex labels in Y (if available), the task of network representation learning is to learn a mapping function $f : v \longmapsto r_v \in \mathbb{R}^d$, where r_v is the learned representation of vertex v, and d is the dimension of the learned representation. The transformation f preserves the original network information, such that two vertices similar in the original network should also be represented similarly in the learned vector space.

The learned vertex representations should satisfy the following conditions: (1) *low-dimensional*, i.e., $d \ll |V|$, in other words, the dimension of learned vertex representations should be much smaller than the dimension of the original adjacency matrix representation for memory efficiency and the scalability of subsequent network analysis tasks; (2) *informative*, i.e., the learned vertex representations should preserve vertex proximity reflected by network structure and vertex attributes and/or vertex labels (if available); (3) *continuous*, i.e., the learned vertex representations should take continuous real values to support subsequent network analytic tasks, like vertex classification, vertex clustering, or anomaly detection, and have smooth decision boundaries to ensure the robustness of these tasks.

Fig. 1 demonstrates a conceptual view of network representation learning, using a toy information network. In this case, only network structure is considered to learn vertex representations. Given an information network shown in Fig. 1(a), the objective of NRL is to embed all network vertices into a low-dimensional space, as depicted in Fig. 1(b). In the embedding space, vertices with structural proximity are represented closely to each other. For example, as vertex 7 and vertex 8 are directly connected, the first-order proximity enforces them close to each other in the embedding space. Though vertex 2 and vertex 5 are not directly connected, they are also embedded closely to each other because they have high second-order proximity, which is reflected by 4 common neighbors shared by these two vertices. Vertex 20 and vertex 25 are not directly connected and neither do they share common direct neighbors. However, they are connected by many k-step paths ($k \geq 3$), which proves that they have high-order proximity. Because



(a) Input: Information Network



(b) Output: Vertex Representations

Fig. 1. A conceptual view of network representation learning. Vertices in (a) are indexed using their ID and are color coded based on their community information. The network representation learning in (b) transforms all vertices into a two dimensional vector space, such that vertices with structural proximity are also close to each other in the new embedding space.

of this, vertex 20 and vertex 25 also have close embeddings. Different from other vertices, vertex 10–16 clearly belong to the same community in the original network. This intra-community proximity guarantees the images of these vertices also exhibit a clear cluster structure in the embedding space.

3 Categorization of Network Representation Learning Techniques

In this section, we propose a taxonomy to categorize existing network representation learning techniques in the current literature, as shown in Fig. 2. The first layer of the taxonomy is based on whether vertex labels are provided for learning. According to this, we categorize network representation learning into two groups: *unsupervised network representation learning* and *semi-supervised network representation learning*.

1) In **unsupervised network representation learning**, no labeled vertices are available for learning vertex representations. In this case, network representation learning is considered as a generic task independent of subsequent learning, and vertex representations are learned in a purely unsupervised manner.
Most of the existing NRL algorithms fall into this category. After vertex representations are learned in

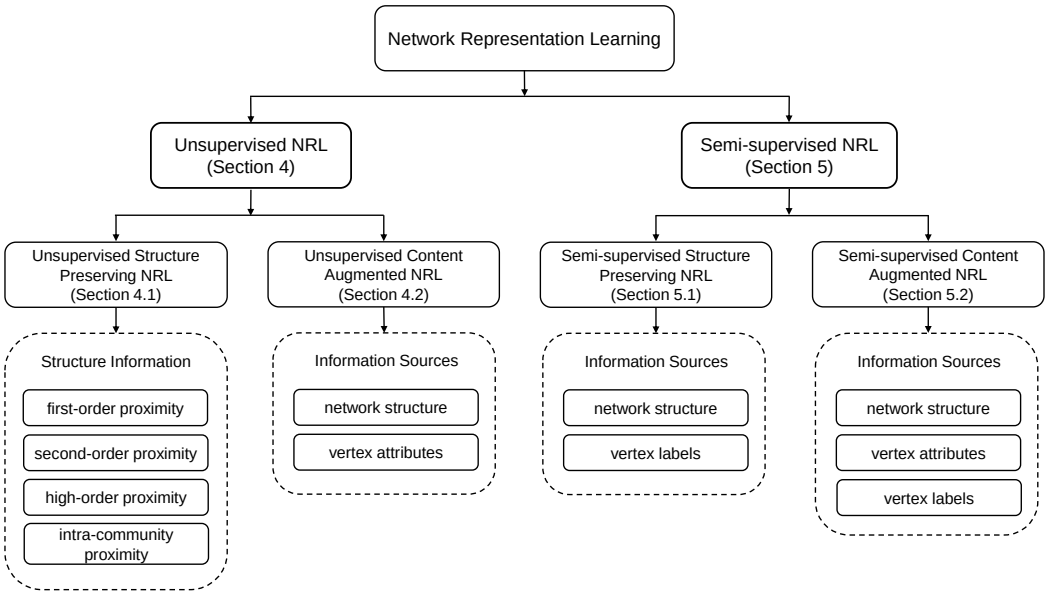


Fig. 2. The proposed taxonomy to summarize network representation learning techniques. We categorize network representation learning into two groups, *unsupervised network representation learning* and *semi-supervised network representation*, depending on whether vertex labels are available for learning. For each group, we further categorize methods into two subgroups, depending on whether the representation learning is based on network topology structure only, or augmented with information from node content.

a new embedded space, they are taken as features to any vector-based algorithms for various learning tasks such as vertex clustering. Unsupervised NRL algorithms can be further divided into two subgroups based on the type of network information available for learning: unsupervised structure preserving methods that preserve only network structure, and unsupervised content augmented methods that incorporate vertex attributes and network structure to learn joint vertex embeddings.

2) In **semi-supervised network representation learning**, there exist some labeled vertices for representation learning. Because vertex labels play an essential role in determining the categorization of each vertex with strong correlations to network structure and vertex attributes, semi-supervised network representation learning is proposed to take advantage of vertex labels available in the network for seeking more effective joint vector representations.
In this setting, network representation learning is coupled with supervised learning tasks such as network vertex classification. A unified objective function is often formulated to simultaneously optimize the learning of vertex representations and the classification of network vertices. Therefore, the learned vertex representations can be both informative and discriminative with respect to different categories. Semi-supervised NRL algorithms can also be categorized into two subgroups, semi-supervised structure preserving methods and semi-supervised content augmented methods, according to the information they preserve.

Table 2 summarizes all NRL algorithms, according to the information sources that they use for representation learning. In general, there are three main types of information sources: network structure, vertex attributes, and vertex labels. From Table 2, we can observe that most NRL algorithms focus on capturing the first-order, the second-order, and the high-order proximity to learn vertex representations, while very few algorithms (e.g., M-NMF [24]) attempt to take advantage of the broader community structure. Most of the unsupervised NRL algorithms only utilize the network structure, and how to leverage vertex attributes to learn vertex representations has not been extensively investigated under this setting. In contrast, under the semi-supervised learning setting, half of algorithms intend to couple vertex attributes with network structure and vertex labels to learn vertex representations.

Approaches to network representation learning in the above two different settings can be summarized into four categories from algorithmic perspectives. Table 3 categorizes the existing NRL algorithms into four groups as follows:

1) **Matrix factorization based methods**. Matrix factorization based methods represent the connections between network vertices in the form of a matrix and use matrix factorization to obtain the embeddings. Different types of matrices are constructed to preserve network structure, such as the k-step transition probability matrix, the modularity matrix, or the vertex-context matrix [7]. By assuming that such high-dimensional vertex representations are only affected by a small quantity of latent factors, matrix factorization is used to embed the high-dimensional vertex representations into a latent, low-dimensional structure preserving space.
Factorization strategies vary across different algorithms according to their objectives. For example, in the Modularity Maximization method [27], eigen

TABLE 2
A summary of NRL algorithms according to the information sources they use for learning

Category	Algorithms	Network Structure				Vertex Attributes	Vertex Labels
		First-order Proximity	Second-order Proximity	Higher-order Proximity	Intra-community Proximity		
Unsupervised	Social Dim. [27], [28], [29]				✓		
	DeepWalk [6]		✓	✓			
	LINE [1]	✓	✓				
	GraRep [22]		✓	✓			
	DNGR [9]		✓	✓			
	SDNE [17]	✓	✓				
	node2vec [30]		✓	✓			
	HOPE [31]		✓	✓			
	APP [32]		✓	✓			
	M-NMF [24]		✓	✓	✓		
	TADW [7]		✓	✓		✓	
	HSCA [8]	✓	✓	✓		✓	
	pRBM [25]	✓				✓	
	UPP-SNE [33]		✓	✓		✓	
Semi-supervised	DDRW [34]		✓	✓			✓
	MMDW [35]		✓	✓			✓
	TLINE [36]	✓	✓				✓
	GENE [37]		✓	✓			✓
	TriDNR [38]		✓	✓		✓	✓
	LDE [39]	✓				✓	✓
	DMF [8]		✓	✓		✓	✓
	Planetoid [40]		✓	✓		✓	✓
	LANE [26]	✓				✓	✓

decomposition is performed on the modularity matrix to learn community indicative vertex representations [41]; in the Text Associated DeepWalk (TADW) algorithm [7], inductive matrix factorization [42] is carried out on the vertex-context matrix to simultaneously preserve vertex textual features and network structure in the learning of vertex representations. Although matrix factorization based NRL algorithms have been proved to be able to learn informative vertex representations, the scalability of these methods is a major bottleneck because carrying out factorization on a matrix with millions of rows and columns is memory intensive and computationally expensive or, sometime, even infeasible.

2) **Random walk based methods**. For scalable vertex representation learning, random walk is exploited to capture structural relationships between vertices. By performing truncated random walks, an information network is transformed into a collection of vertex sequences, in which, the occurrence frequency of a vertex-context pair measures the structural distance between them. Borrowing the idea of word representation learning [43], [44], vertex representations are then learned by using each vertex to predict its contexts. DeepWalk [6] is the pioneer work in using random walks to learn vertex representations. Following this line, node2vec [30] exploits a biased random walk strategy to capture more global structure.

As the extensions of the structure preserving only version, algorithms like DDRW [34] and GENE [37] incorporate vertex labels with network structure to harness representation learning, and Tri-DNR [38] enforces the model with both vertex labels and attributes. As these models can be trained in an on-line manner, they are able to scale to large-scale information networks.

3) **Edge modeling based methods**. Different from the NRL algorithms that use matrix or random walk to capture network structure, the edge modeling based methods directly learn the structure preserving vertex representations from vertex-vertex connections. For capturing the first-order and second-order proximity, LINE [1] models a joint probability distribution and a conditional probability distribution, respectively, on connected vertices. For learning the representations of linked documents, LDE [39] models the document-document relationships by maximizing the conditional probability between connected documents. pRBM [25] adapts the RBM [45] model to linked data by making the hidden RBM representations of connected vertices similar to each other.

TABLE 3
A categorization of NRL algorithms from methodology perspectives

Methodology	Algorithms	Time Complexity	Optimization Method
Matrix Factorization	Social Dim. [27], [28]	$O(d\|V\|^2)$	Eigen Decomposition
	GraRep [22]	$O(\|V\|\|E\| + d\|V\|^2)$	
	HOPE [31]	$O(d^2 I\|E\|)$	
	M-NMF [24]	$O(dI\|V\|^2)$	Alternative Optimization
	TADW [7]	$O(\|V\|\|E\| + dI\|E\| + dmI\|V\| + d^2 I\|V\|)$	
	HSCA [8]	$O(\|V\|\|E\| + dI\|E\| + dmI\|V\| + d^2 I\|V\|)$	
	MMDW [35]		
	DMF [8]	$O(\|V\|\|E\| + dI\|E\| + dmI\|V\| + d^2 I\|V\|)$	
	LANE [26]	$O(m\|V\|^2 + dI\|V\|^2)$	
Random Walk	UPP-SNE [33]	$O(I\|E\| \cdot nnz(X))$	Gradient Descent
	DeepWalk [6]	$O(d\|V\| \log \|V\|)$	Stochastic Gradient Descent
	node2vec [30]	$O(d\|V\|)$	
	APP [32]	$O(d\|V\|)$	
	DDRW [34]	$O(d\|V\| \log \|V\|)$	
	GENE [37]		
	TriDNR [38]	$O(d \cdot nnz(X) \log(m) + d\|V\| \log(\|V\|)$	
	Planetoid [40]		
Edge Modeling	LINE [1]	$O(d\|E\|)$	
	TLINE [36]	$O(d\|E\|)$	
	LDE [39]	$O(dI \cdot nnz(X) + dI\|E\| + dI\|\mathcal{Y}\|\|V\|)$	
	pRBM [25]	$O(dmI\|V\|)$	
Deep Learning	DNGR [9]		
	SDNE [17]	$O(dI\|E\|)$	

4) **Deep learning based methods**. To extract complex structure features and learn deep, highly nonlinear vertex representations, the deep learning techniques [46], [47] are also applied to NRL. To learn deep low-dimensional vertex representations, DNGR [9] applies the *stacked denoising autoencoders* (SDAE) [47] to the high-dimensional positive pointwise mutual information (PPMI) matrix representations. Another deep learning based NRL algorithm is SDNE [17]. SDNE is realized by a semi-supervised deep autoencoder model [46], in which the unsupervised component reconstructs the second-order proximity to retain the global network structure, while the supervised component exploits the first-order proximity as supervised information to preserve the local network structure. Deep learning based methods have the ability to capture nonlinearity in networks, but their scalability and interpretability need to be further investigated.

For these NRL algorithms, Table 3 provides a detailed analysis of their time complexity and optimization method used. To analyze the time complexity, we introduce a new notation I to represent the number of iterations and we use $nnz(\cdot)$ to denote the number of non-zero entries of a matrix. Here, the time complexity of MMDW, GENE, Planetoid and DNGR is not given due to the lack of technique details in the original papers. In a nutshell, four kinds of solutions are used to optimize the objectives of the existing NRL algorithms: (1) **eigen decomposition** that involves finding top-d eigenvectors of a matrix, (2) **alternative optimization** that optimizes one variable with the remaining variables fixed alternately, (3) **gradient descent** that updates all parameters at each iteration for optimizing the overall objective, and (4) **stochastic gradient descent** that optimizes the partial objective stochastically in an on-line mode.

The time complexity comparison in Table 3 shows that matrix factorization based approaches usually require quadratic time complexity with respect to number of vertices. This prevents them from scaling to large datasets. Random walk, edge modeling, and deep learning based methods that mainly adopt stochastic gradient descent optimization strategy are much more efficient than matrix factorization based methods that are solved by eigen decomposition and alternative optimization.

4 Unsupervised Network Representation Learning

In this section, we review unsupervised network representation learning methods by separating them into two subsections, as outlined in Fig. 2. After that, we summarize key characteristics of the methods and compare their differences across the two categories.

4.1 Unsupervised Structure Preserving Network Representation Learning

Structure preserving network representation learning refers to methods that intend to preserve network structure, in

the sense that vertices close to each other in the original network space should be represented similarly in the new embedded space. In this category, research efforts have been focused on designing various models to capture structure information conveyed by the original network as much as possible. In the literature, network structure is mainly considered at two levels: the local structure, e.g., the proximity between connected vertices, and the global structure, e.g., the community structure in the network.

4.1.1 Learning Latent Social Dimensions

The social dimension based network representation learning algorithms try to construct social actors' embeddings through their membership or affiliation to a number of social dimensions. To infer these latent social dimensions, the phenomenon of "community" in social networks is considered, stating that social actors sharing similar properties often form groups with denser within-group connections. Thus, the problem boils down to one classical network analytic task—community detection—that aims to discover a set of communities with denser within-group connections than between-group connections. Three clustering techniques, including Modularity Maximization [27], Spectral Clustering [28] and Edge Clustering [29] are employed to discover latent social dimensions. Each social dimension describes the likelihood of a vertex belonging to a plausible affiliation. Social dimension based methods intend to preserve the global community structure, but neglect local structure properties, e.g., the first-order and second-order proximity.

4.1.2 DeepWalk

DeepWalk [6] generalizes the idea of the Skip-Gram model [43], [44] that utilizes word context in sentences to learn latent representations of words, to the learning of latent vertex representations in networks, by making an analogy between natural language sentence and short random walk sequence. Given a random walk sequence with length L, $\{v_1, v_2, \cdots, v_L\}$, following Skip-Gram, DeepWalk learns the representation of vertex v_i by using it to predict its context vertices, which is achieved by the optimization problem:

$$\min_{f} \ -\log \Pr(\{v_{i-t}, \cdots, v_{i+t}\} \setminus v_i | f(v_i)), \tag{1}$$

where $\{v_{i-t}, \cdots, v_{i+t}\} \setminus v_i$ are the context vertices of vertex v_i within t window size. Making conditional independence assumption, the probability $\Pr(\{v_{i-t}, \cdots, v_{i+t}\} \setminus v_i | f(v_i))$ is approximated as

$$\Pr\left(\{v_{i-t}, \cdots, v_{i+t}\} \setminus v_i | f(v_i)\right) = \prod_{j=i-t, j\neq i}^{i+t} \Pr(v_j | f(v_i)). \tag{2}$$

Following the DeepWalk's learning architecture, vertices that share similar context vertices in random walk sequences are supposed to be represented closely in the new embedding space. Considering the fact that context vertices in random walk sequences describe neighborhood structure, DeepWalk actually represents vertices sharing similar neighbors (direct or indirect) closely in the embedding space and the second-order and high-order proximity is preserved.

4.1.3 Large-scale Information Network Embedding (LINE)

Instead of exploiting random walks to capture network structure, LINE [1] learns network vertex representations by explicitly modeling the first-order and the second-order proximity. To preserve the first-order proximity, LINE minimizes the following objective:

$$O_1 = d(\hat{p}_1(\cdot,\cdot), p_1(\cdot,\cdot)), \tag{3}$$

Above, for each vertex pair v_i and v_j with $(v_i, v_j) \in E$, $p_1(\cdot,\cdot)$ is the joint distribution modeled by their latent embeddings r_{v_i} and r_{v_j}, $\hat{p}_1(v_i, v_j)$ is the empirical distribution between them, and $d(\cdot,\cdot)$ is the distance between the two distributions.

To preserve the second-order proximity, LINE minimizes the following objective:

$$O_2 = \sum_{v_i \in V} \lambda_i d(\hat{p}_2(\cdot|v_i), p_2(\cdot|v_i)), \tag{4}$$

where $p_2(\cdot|v_i)$ is the context conditional distribution for each $v_i \in V$ modeled by vertex embeddings, $\hat{p}_2(\cdot|v_i)$ is the empirical conditional distribution and λ_i is the prestige of vertex v_i. Here, vertex context is determined by its neighbors, i.e., for each v_j, v_j is v_i's context, if and only if $(v_i, v_j) \in E$.

By minimizing these two objectives, LINE learns two kinds of vertex representations that preserve the first-order and the second-order proximity, respectively. The concatenation of these two vertex representations are outputted as the final vertex representation.

4.1.4 GraRep

Following the idea of DeepWalk [6], GraRep [22] extends the skip-gram model to capture the high-order proximity, i.e., vertices sharing common k-step neighbors $(k \geq 1)$ should have similar latent representations. Specifically, for each vertex, GraRep defines its k-step neighbors $(k \geq 1)$ as context vertices, and for each $1 \leq k \leq K$, to learn k-step vertex representations, GraRep employs the matrix factorization version of skip-gram:

$$\left[U^k, \Sigma^k, V^k\right] = SVD(X^k). \tag{5}$$

where X^k is the log k-step transition probability matrix. The k-step representation for vertex v_i is constructed as the ith row of the matrix $U_d^k(\Sigma_d^k)^{\frac{1}{2}}$, where U_d^k is the first-d columns of U^k and Σ_d^k is the diagonal matrix composed of the top d singular values. After different k-step vertex representations are learned, GraRep concatenates them together as the final vertex representations.

4.1.5 Deep Neural Networks for Graph Representations (DNGR)

To overcome the weakness of truncated random walks in exploiting vertex contextual information, i.e., the difficulty in capturing correct contextual information for vertices at the boundary of sequences and the difficulty in determining the walk length and the number of walks, DNGR [9] utilizes the random surfing model to capture the contextual relatedness between each pair of vertices and preserves them into $|V|$-dimensional vertex representations X.

To extract complex features and model non-linearities, DNGR applies the *stacked denoising autoencoders* (SDAE) [47] to the high-dimensional vertex representations X to learn deep low-dimensional vertex representations.

4.1.6 Structural Deep Network Embedding (SDNE)

SDNE [17] is a deep learning based approach for network representation learning that uses a semi-supervised deep autoencoder model to capture non-linearity in network structure. In the unsupervised component, SDNE learns the second-order proximity preserving vertex representations via reconstructing the $|V|-$dimensional vertex adjacent matrix representations. In the supervised part, SDNE imports the first-order proximity by penalizing the distance of connected vertices in the embedding space.

The minimization objective of the unsupervised component is

$$\mathcal{L}_{2nd} = \sum_{i=1}^{|V|} \|(r_{v_i}^{(0)} - \hat{r}_{v_i}^{(0)}) \odot \boldsymbol{b}_i\|, \tag{6}$$

where $r_{v_i}^{(0)} = S_{i:}$ is the input representation and $\hat{r}_{v_i}^{(0)}$ is the reconstructed representation. In the above objective, $\odot$ means the Hadamard product, $\boldsymbol{b}_i = \{b_{ij}\}_{j=1}^{|V|}$, with $b_{ij} = 1$ for $S_{ij} = 0$ and $b_{ij} = \beta > 1$ for $S_{ij} \neq 0$.

In the supervised component, to enforce the first-order proximity in the embedding space, the distance between connected vertices is penalized. The loss function for this objective is defined as:

$$\mathcal{L}_{1st} = \sum_{i,j=1}^{|V|} S_{ij} \|r_{v_i}^{(K)} - r_{v_j}^{(K)}\|_2^2, \tag{7}$$

where $r_{v_i}^{(K)}$ is the K-th layer representation of vertex v_i, with K being the number of hidden layers.

To preserve the first-order and second-order proximity, SDNE minimizes the joint objective function:

$$\mathcal{L} = \mathcal{L}_{2nd} + \alpha \mathcal{L}_{1st} + \nu \mathcal{L}_{reg}, \tag{8}$$

where $\mathcal{L}_{reg}$ is the regularization term to prevent overfitting. After the minimization of the objective (8) is solved, for each vertex v_i, the learned K-th layer representation $r_{v_i}^{(K)}$ is taken as the final representation r_{v_i}.

4.1.7 node2vec

In contrast to the rigid strategy of defining neighborhood (context) for each vertex, node2vec [30] designs a flexible neighborhood sampling strategy, i.e., biased random walk, which smoothly interpolates between two extreme sampling strategies, i.e., Breadth-first Sampling (BFS) and Depth-first Sampling (DFS). The biased random walk exploited in node2vec can better preserve both the local structure (the second-order proximity) and the global structure (the high-order proximity).

Following the skip-gram architecture, given the set of neighborhood vertices $N(v_i)$ generated by biased random walk, node2vec learns the vertex representation $f(v_i)$ by optimizing the occurrence probability of neighbor vertices $N(v_i)$ conditioned on the representation of vertex v_i, $f(v_i)$:

$$\max_f \sum_{v_i \in V} \log \Pr(N(v_i)|f(v_i)). \tag{9}$$

4.1.8 High-order Proximity Preserved Embedding (HOPE)

HOPE [31] learns vertex representations that capture the asymmetric high-order proximity in directed networks. In undirected networks, the transitivity is symmetric, but it is asymmetric in directed networks. For example, in an directed information network, if there is a directed link from vertex v_i to vertex v_j and a directed link from vertex v_j to vertex v_k, it is more likely to have a directed link from v_i to v_k, but not from v_k to v_i.

To preserve the asymmetric transitivity, HOPE [31] learns two vertex embedding vectors $U^s, U^t \in \mathbb{R}^{|V| \times d}$, which is called source and target embedding vectors, respectively. After constructing the high-order proximity matrix S from four proximity measures, Katz Index [48], Rooted PageRank [49], Common Neighbors and Adamic-Adar. The asymmetric high-order proximity preserving vertex embedding is learned by solving the following matrix factorization problem:

$$\min_{U_s, U_t} \|S - U^s \cdot U^{t^{\mathrm{T}}}\|_F^2. \tag{10}$$

4.1.9 Asymmetric Proximity Preserving graph embedding (APP)

APP [32] is another NRL algorithm designed to capture asymmetric proximity, by using a Monte Carlo approach to approximate the asymmetric Rooted PageRank proximity [49]. Similar to HOPE, APP has two representations for each vertex v_i, the one as a source role $r_{v_i}^s$ and the other as a target role $r_{v_i}^t$. For each sampled path starting from v_i and ending with v_j, the representations are learned by maximizing the target vertex v_j occurrence probability conditioned on the source vertex v_i:

$$\Pr(v_j|v_i) = \frac{\exp(r_{v_i}^s \cdot r_{v_j}^t)}{\sum_{v \in V} \exp(r_{v_i}^s \cdot r_v^t)}. \tag{11}$$

4.1.10 Modularized Nonnegative Matrix Factorization (M-NMF)

The majority of above NRL algorithms focus on preserving the local structure, like the first-order, second-order and high-order proximity between network vertices. M-NMF [24] augments the second-order and high-order proximity with broader community structure to learn more informative vertex embeddings $U \in \mathbb{R}^{|V| \times d}$ using the following objective:

$$\begin{aligned} &\min_{M,U,H,C} \|S - MU^{\mathrm{T}}\|_F^2 + \alpha \|H - UC^{\mathrm{T}}\|_F^2 - \beta tr(H^{\mathrm{T}} B H) \\ &s.t., \quad M \geq 0, U \geq 0, H \geq 0, C \geq 0, tr(H^{\mathrm{T}} H) = |V|, \end{aligned} \tag{12}$$

where $\alpha > 0$ and $\beta > 0$ are two parameters that control the contribution of respective terms.

In the objective of (12), the second-order and high-order proximity preserving vertex embedding U is learned by minimizing $\|S - MU^{\mathrm{T}}\|_F^2$, with $S \in \mathbb{R}^{|V| \times |V|}$ being the first-order and the second-order combined vertex pairwise proximity matrix, which in fact captures the second-order and the high-order proximity when taken as representations. The community indicative vertex embedding H is learned by maximizing $tr(H^{\mathrm{T}} B H)$, which is actually the objective of modularity maximization with B being the modularity matrix. The minimization on $\|H - UC^{\mathrm{T}}\|_F^2$ makes these

two embeddings consistent with each other by importing a community representation matrix C.

Summary: Most of the above unsupervised structure preserving NRL algorithms are matrix factorization based methods and random walk based methods. The essence of matrix factorization based methods (i.e., Social dimensions [27], [28], [29], GraRep [22], HOPE [31], M-NMF [24]) performing linear transformation on the original $|V|$-dimensional space that exhibits structure information, while deep learning based methods (DNGR [9] and SDNE [17]) use deep neural networks to realize non-linear transformation. These methods usually involve performing operation on the $|V| \times |V|$ scale matrix, making it hard to scale up. On the other hand, random walk based methods (i.e., DeepWalk [6], node2vec [30], APP [32]) exploits random walk to preserve local structure and they are easy to parallelize.

4.2 Unsupervised Content Augmented Network Representation Learning

Besides network structure, real-world networks are often attached with content information as vertex attributes, such as Webpages in Webpage networks, papers in citation networks, and user metadata in social networks. Vertex attributes provide direct evidence to measure content-level similarity between vertices. Therefore, network representation learning can be significantly improved if vertex attribute information is properly incorporated into the learning process. Recently, several content augmented NRL algorithms have been proposed to incorporate network structure and vertex attributes to reinforce the network representation learning.

4.2.1 Text-Associated DeepWalk (TADW)

TADW [7] fistly proves the equivalence between DeepWalk [6] and the following matrix factorization:

$$\min_{W,H} \|M - W^{\mathrm{T}} H\|_F^2 + \frac{\lambda}{2}(\|W\|_F^2 + \|H\|_F^2), \tag{13}$$

where W and H are learned latent embeddings and M is the node-context matrix carrying transition probability between each node pair within k steps. Then, textual features are imported through inductive matrix factorization [42]

$$\min_{W,H} \|M - W^{\mathrm{T}} HT\|_F^2 + \frac{\lambda}{2}(\|W\|_F^2 + \|H\|_F^2), \tag{14}$$

where T is vertex content textual feature matrix. After the above optimization problem is solved, the final vertex representations are formed by taking the concatenation of W and HT.

4.2.2 Homophily, Structure, and Content Augmented Network Representation Learning (HSCA)

Despite its ability to incorporate content textural features, TADW [7] only considers structural context of network vertices, i.e., the second-order and high-order proximity, but ignores the important homophily property (the first-order proximity) in its learning framework. Based on this, HSCA [18] is proposed to simultaneously integrates homophily, structural context, and vertex content to learn effective network representations.

For TADW, the learned representation for the i-th vertex v_i is $\left[W_{i:}^{\mathrm{T}}, (HT_{i:})^{\mathrm{T}}\right]^{\mathrm{T}}$, where $W_{i:}$ and $T_{i:}$ is the i-th row of W and T, respectively. To enforce the first-order proximity, HSCA introduces a regularization term that penalizes the distance between connected vertices in the embedding space, which is formulated as

$$\mathcal{R}(W,H) = \frac{1}{4}\sum_{i,j=1}^{|V|} S_{ij} \| \begin{bmatrix} W_{i:} \\ HT_{i:} \end{bmatrix} - \begin{bmatrix} W_{j:} \\ HT_{j:} \end{bmatrix} \|_2^2 \tag{15}$$

where S is the adjacent matrix. The final objective of HSCA is

$$\min_{W,H} \|M - W^{\mathrm{T}} HT\|_F^2 + \frac{\lambda}{2}(\|W\|_F^2 + \|H\|_F^2) + \mu\mathcal{R}(W,H), \tag{16}$$

where λ and μ are the trade-off parameters. After solving the above optimization problem, the concatenation of W and HT is taken as the final vertex representations.

4.2.3 Paired Restricted Boltzmann Machine (pRBM)

By leveraging the strength of Restricted Boltzmann Machine (RBM) [45], [25] designs a novel model called Paired RBM (pRBM) to learn vertex representations by combining vertex attributes and link information. The pRBM considers the networks with vertices associated with binary attributes. For each edge $(v_i, v_j) \in E$, the attributes for v_i and v_j are $\mathbf{v}^{(i)}$ and $\mathbf{v}^{(j)} \in \{0,1\}^m$, and their hidden representations are $\mathbf{h}^{(i)}$ and $\mathbf{h}^{(j)} \in \{0,1\}^d$. Vertex hidden representations are learned by maximizing the joint probability of pRBM defined over $\mathbf{v}^{(i)}$, $\mathbf{v}^{(j)}$, $\mathbf{h}^{(i)}$ and $\mathbf{h}^{(j)}$:

$$\begin{aligned} &\Pr(\mathbf{v}^{(i)}, \mathbf{v}^{(j)}, \mathbf{h}^{(i)}, \mathbf{h}^{(j)}, w_{ij}; \boldsymbol{\theta}) = \\ &\exp(-E(\mathbf{v}^{(i)}, \mathbf{v}^{(j)}, \mathbf{h}^{(i)}, \mathbf{h}^{(j)}, w_{ij}))/Z, \end{aligned} \tag{17}$$

where $\boldsymbol{\theta} = \{\mathbf{W} \in \mathbb{R}^{d\times m}, \mathbf{b} \in \mathbb{R}^{d\times 1}, \mathbf{c} \in \mathbb{R}^{m\times 1}, \mathbf{M} \in \mathbb{R}^{d\times d}\}$ is the parameter set and Z is the normalization term. To model the joint probability, the energy function is defined as

$$\begin{aligned} &E(\mathbf{v}^{(i)}, \mathbf{v}^{(j)}, \mathbf{h}^{(i)}, \mathbf{h}^{(j)}, w_{ij}) = \\ &- w_{ij}(\mathbf{h}^{(i)})^{\mathrm{T}}\mathbf{M}\mathbf{h}^{(j)} - (\mathbf{h}^{(i)})^{\mathrm{T}}\mathbf{W}\mathbf{v}^{(i)} - \mathbf{c}^{\mathrm{T}}\mathbf{v}^{(i)} - \mathbf{b}^{\mathrm{T}}\mathbf{h}^{(i)} \\ &- (\mathbf{h}^{(j)})^{\mathrm{T}}\mathbf{W}\mathbf{v}^{(j)} - \mathbf{c}^{\mathrm{T}}\mathbf{v}^{(i)} - \mathbf{b}^{\mathrm{T}}\mathbf{h}^{(j)}, \end{aligned} \tag{18}$$

where $w_{ij}(\mathbf{h}^{(i)})^{\mathrm{T}}\mathbf{M}\mathbf{h}^{(j)}$ forces the latent representations of v_i and v_j to be close and w_{ij} is the weight of edge (v_i, v_j).

4.3 User Profile Preserving Social Network Embedding (UPP-SNE)

UPP-SNE [33] aims to leverage user profile features to enhance the embedding learning of users in social networks. Compared with the vertex textural content features, user profiles has two unique properties: (1) user profiles are noise, sparse and incomplete and (2) different dimensions of user profile features are not topic-consistent. To filter out noisy information and take advantage of useful information in user profiles, UPP-SNE constructs user representations by performing a non-linear mapping on user profile features, which is supervised by network structure.

The approximated kernel mapping [50] is used in UPP-SNE to construct user embedding from user profile features:

$$f(v_i) = \varphi(\boldsymbol{x}_i) = \frac{1}{\sqrt{d}}\Big[\cos(\boldsymbol{\mu}_1^{\mathrm{T}}\boldsymbol{x}_i), \cdots, \cos(\boldsymbol{\mu}_d^{\mathrm{T}}\boldsymbol{x}_i), \sin(\boldsymbol{\mu}_1^{\mathrm{T}}\boldsymbol{x}_i), \cdots, \sin(\boldsymbol{\mu}_d^{\mathrm{T}}\boldsymbol{x}_i)\Big]^{\mathrm{T}}, \tag{19}$$

where $\boldsymbol{x}_i$ is the user profile feature vector of vertex v_i and $\boldsymbol{\mu}_i$ is the corresponding coefficient vector.

To supervise the learning of the non-linear mapping and make user profiles and network structure complement each other, the objective of DeepWalk [6] is used:

$$\min_{f} \; -\log \Pr(\{v_{i-t}, \cdots, v_{i+t}\} \setminus v_i | f(v_i)), \tag{20}$$

where $\{v_{i-t}, \cdots, v_{i+t}\} \setminus v_i$ is the context vertices of vertex v_i within w-window size in the given random walk sequence.

Summary: The above unsupervised content augmented NRL algorithms incorporate vertex content features in two ways. The first, used by TADW [7] and HSCA [18], is to couple the network structure with vertex content features via inductive matrix factorization [42]. This process can be considered as a linear transformation on vertex attributes constrained by network structure. The second is to perform a non-linear mapping to construct new vertex embeddings that respect network structure. For example, RBM [45] and the approximated kernel mapping [50] is used by pRBM [25] and UPP-SNE [33], respectively, to achieve this goal.

5 Semi-Supervised Network Representation Learning

Label information attached with vertices directly indicates vertices' group or class affiliation. Such labels have strong correlations, although not always consistent, to network structure and vertex attributes, and are always helpful in learning informative and discriminative network representations. Semi-supervised NRL algorithms are developed along this line to make use of vertex labels available in the network for seeking more effective vertex representations.

5.1 Semi-Supervised Structure Preserving NRL

Built upon unsupervised structure preserving NRL algorithms (Section 4.1), the first group of semi-supervised NRL algorithms are proposed to simultaneously optimize the representation learning and discriminative learning. As a result, the information derived from vertex labels can help improve the representative and discriminative power of the learned vertex representations.

5.1.1 Discriminative Deep Random Walk (DDRW)

Inspired by the discriminative representation learning [51], [52], DDRW [34] proposes to learn discriminative network representations through jointly optimizing the objective of DeepWalk [6] together with the following L2-loss Support Vector Classification classification objective:

$$\mathcal{L}_c = C\sum_{i=1}^{|V|}(\sigma(1 - Y_{ik}\beta^{\mathrm{T}} r_{v_i}))^2 + \frac{1}{2}\beta^{\mathrm{T}}\beta, \tag{21}$$

where $\sigma(x)$ equals to x, if $x > 0$ and otherwise $\sigma(x)$ equals to 0.

The joint objective of DDRW is thus defined as

$$\mathcal{L} = \eta\mathcal{L}_{DW} + \mathcal{L}_c, \tag{22}$$

where $\mathcal{L}_{DW}$ is the objective function of DeepWalk. The objective (22) aims to learn discriminative vertex representations for binary classification for the k-th class. DDRW is generalized to handle multi-class classification by using the *one-against-rest* strategy [53].

5.1.2 Max-Margin DeepWalk (MMDW)

Similarly, MMDW [35] couples the objective of the matrix factorization version DeepWalk [7] with the following multi-class Support Vector Machine objective with $\{(r_{v_1}, Y_{1:}), \cdots, (r_{v_T}, Y_{T:})\}$ training set:

$$\begin{aligned} &\min_{W,\xi} \mathcal{L}_{SVM} = \min_{W,\xi} \frac{1}{2}\|W\|_2^2 + C\sum_{i=1}^{T}\xi_i, \\ &s.t.\ \mathrm{w}_{l_i}^{\mathrm{T}} r_{v_i} - \mathrm{w}_j^{\mathrm{T}} r_{v_i} \geq e_i^j - \xi_i,\ \forall i, j, \end{aligned} \tag{23}$$

where

$$e_i^j = \begin{cases} 1, \text{ if } Y_{ij} = -1, \\ 0, \text{ if } Y_{ij} = 1. \end{cases} \tag{24}$$

Here, $W = [\mathrm{w}_1, \cdots, \mathrm{w}_m]^{\mathrm{T}}$ is the weight matrix of SVM, $l_i = j$ for $Y_{ij} = 1$ and $\xi = [\xi_1, \cdots, \xi_T]$ is the slack variable that tolerates classification errors in the training set.

The joint objective of MMDW is

$$\begin{aligned} &\min_{U,H,W,\xi} \mathcal{L} = \min_{U,H,W,\xi} \mathcal{L}_{DW} + \frac{1}{2}\|W\|_2^2 + C\sum_{i=1}^{T}\xi_i, \\ &s.t.\ \mathrm{w}_{l_i}^{\mathrm{T}} r_{v_i} - \mathrm{w}_j^{\mathrm{T}} r_{v_i} \geq e_i^j - \xi_i,\ \forall i, j. \end{aligned} \tag{25}$$

where $\mathcal{L}_{DW}$ is the objective of the matrix factorization version of DeepWalk.

5.1.3 Transductive LINE (TLINE)

Along similar lines, TLINE [36] is proposed as a semi-supervised extension of LINE [1] that simultaneously learns LINE's vertex representations and an SVM classifier. Let $\{v_1, v_2, \cdots, v_L\}$ and $\{v_{L+1}, \cdots, v_{|V|}\}$ denote the set of labeled and unlabeled vertices, respectively. TLINE trains an multi-class SVM classifier on $\{v_1, v_2, \cdots, v_L\}$ by optimizing the objective:

$$\mathcal{O}_{svm} = \sum_{i=1}^{L}\sum_{k=1}^{K}\max(0, 1 - Y_{ik}\mathrm{w}_k{}^{\mathrm{T}} r_{v_i}) + \lambda\|\mathrm{w}_k\|_2^2. \tag{26}$$

Based on LINE's formulations that preserve the first-order and second-order proximity, TLINE optimizes two objective functions:

$$\mathcal{O}_{TLINE(1st)} = \mathcal{O}_{line1} + \beta\mathcal{O}_{svm}, \tag{27}$$

$$\mathcal{O}_{TLINE(2nd)} = \mathcal{O}_{line2} + \beta\mathcal{O}_{svm}, \tag{28}$$

where β is the trade-off parameter. Borrowing the power of LINE for dealing with large-scale networks, TLINE is claimed to be able to learn discriminative vertex representations for large-scale networks with very low time and memory cost.

5.1.4 Group Enhanced Network Embedding (GENE)

To learn discriminative vertex representations, GENE [37] integrates group (label) information with network structure in a probabilistic manner. GENE assumes that vertices should be embedded closely in low-dimensional space, if they share similar neighbors or join similar groups. Inspired by DeepWalk [6] and document modeling [54], [55], the mechanism of GENE for learning group label informed vertex representations is achieved by maximizing the following log probability:

$$\mathscr{L} = \sum_{g_i \in \mathcal{Y}} \Bigg[\alpha \sum_{W \in W_{g_i}} \sum_{v_j \in W} \log \Pr(v_j | v_{j-t}, \cdots, v_{j+t}, g_i) + \beta \sum_{\hat{v}_j \in \hat{W}_{g_j}} \log \Pr(\hat{v}_j | g_i) \Bigg], \tag{29}$$

where $\mathcal{Y}$ is the set of different groups, W_{g_i} is the set of random walk sequences labeled with g_i, $\hat{W}_{g_i}$ is the set of vertices randomly sampled from group g_i, α and β are the trade-off parameters.

Summary: In general, two strategies are adopted by the existing semi-supervised structure preserving NRL algorithms to empower vertex labels: the first (i.e., DDRW [34], MMDW [35] and TLINE [36]) is to enforce a classification loss on the objective function of representation learning, and the second (used by GENE [37]) is to maximize the conditional probability of co-occurring vertices given vertex labels. Among others, MMDW [35] suffers from the same scalability problem as other matrix factorization based methods.

5.2 Semi-Supervised Content Augmented NRL

Recently, more research efforts have shifted to the development of label and content augmented NRL algorithms that investigate the use of vertex content and labels to assist with network representation learning. With content information incorporated, the learned vertex representations are expected to be more informative, and with label information considered, the learned vertex representations can be highly customized for the underlying classification task.

5.2.1 Tri-Party Deep Network Representation (TriDNR)

Using a coupled neural network framework, TriDNR [38] learns vertex representations from three information sources: network structure, vertex content and vertex labels. To capture the vertex content and label information, TriDNR adapts the Paragraph Vector model [54] to describe the vertex-word correlation and the label-word correspondence by maximizing the following objective:

$$\mathcal{L}_{PV} = \sum_{i \in L} \log \Pr(w_{-b} : w_b | c_i) + \sum_{i=1}^{|V|} \log \Pr(w_{-b} : w_b | v_i), \tag{30}$$

where $\{w_{-b} : w_b\}$ is a sequence of words inside a contextual window of length $2b$, c_i is the class label of vertex v_i, and L is the set of indices of labeled vertices.

TriDNR is then realized by coupling the Paragraph Vector objective with DeepWalk objective:

$$\max\ (1-\alpha)\mathcal{L}_{DW} + \alpha \mathcal{L}_{PV}, \tag{31}$$

where $\mathcal{L}_{DW}$ is the DeepWalk maximization objective function and α is the trade-off parameter.

5.2.2 Linked Document Embedding (LDE)

LDE [39] is proposed to learn representations for linked documents, which are actually the vertices of citation or webpage networks. Similar to TriDNR [38], LDE learns vertex representations by modeling three kinds of relations, i.e., word-word-document relations, document-document relations, and document-label relations. LDE is realized by solving the following optimization problem:

$$\begin{aligned} \min_{\boldsymbol{W},\boldsymbol{D},\boldsymbol{Y}} & -\frac{1}{|\mathcal{P}|} \sum_{(w_i,w_j,d_k)\in\mathcal{P}} \log \Pr(w_j | w_i, d_k) \\ & -\frac{1}{|E|} \sum_i \sum_{j:(v_i,v_j)\in E} \log \Pr(d_j | d_i) \\ & -\frac{1}{|\mathcal{Y}|} \sum_{i:y_i \in \mathcal{Y}} \log \Pr(y_i | d_i) \\ & + \gamma(\|\boldsymbol{W}\|_F^2 + \|\boldsymbol{D}\|_F^2 + \|\boldsymbol{Y}\|_F^2). \end{aligned} \tag{32}$$

Here, the probability $\Pr(w_j|w_i, d_k)$ is used to model word-word-document relations, which means the probability that in document d_k, word w_j is a neighboring word of w_i. To capture word-word-document relations, triplets (w_i, w_j, d_k) are extracted, with the word-neighbor pair (w_i, w_j) occurring in document d_k. The set of triplets (w_i, w_j, d_k) is denoted by $\mathcal{P}$. The document-document relations are captured by the conditional probability between linked document pairs (d_i, d_j), $\Pr(d_j|d_i)$. The document-label relations are also considered by modeling $\Pr(y_i|d_i)$, the probability for the occurrence of class label y_i conditioned on document d_i. In (32), $\boldsymbol{W}$, $\boldsymbol{D}$ and $\boldsymbol{Y}$ is the embedding matrix for words, documents and labels, respectively. To avoid overfitting, a regularization term $\|\boldsymbol{W}\|_F^2 + \|\boldsymbol{D}\|_F^2 + \|\boldsymbol{Y}\|_F^2$ is added.

5.2.3 Discriminative Matrix Factorization (DMF)

To empower vertex representations with discriminative ability, [8] enforces the objective of TADW (14) with an empirical loss minimization of a linear classifier trained on labeled vertices:

$$\begin{aligned} \min_{W,H,\boldsymbol{\eta}} & \frac{1}{2} \sum_{i,j=1}^{|V|} (M_{ij} - \boldsymbol{w}_i^{\mathrm{T}} H \boldsymbol{t}_j)^2 + \frac{\mu}{2} \sum_{n \in \mathcal{L}} (Y_{n1} - \boldsymbol{\eta}^{\mathrm{T}} \boldsymbol{x}_n)^2 \\ & + \frac{\lambda_1}{2}(\|H\|_F^2 + \|\boldsymbol{\eta}\|_2^2) + \frac{\lambda_2}{2}\|W\|_F^2, \end{aligned} \tag{33}$$

where $\boldsymbol{w}_i$ is the i-th column of vertex representation matrix W and $\boldsymbol{t}_j$ is j-th column of vertex textual feature matrix T, and $\mathcal{L}$ is the set of indices of labeled vertices. DMF considers binary-class classification, i.e. $\mathcal{Y} = \{+1, -1\}$. Hence, Y_{n1} is used to denote the class label of vertex v_n.

DMF constructs vertex representations from W rather that W and HT, which is based on empirical findings that W is informative enough to act as vertex representations. In the objective of (33), $\boldsymbol{x}_n$ is set to $[\boldsymbol{w}_n^{\mathrm{T}}, 1]^{\mathrm{T}}$, which incorporate

the intercept term b of the linear classifier into $\boldsymbol{\eta}$. The optimization problem (33) is solved by optimizing W, H and $\boldsymbol{\eta}$ alternately. After the optimization problem is solved, the discriminative and informative vertex representations together with the linear classifier are learned, which can work together for classifying unlabeled vertices in networks.

5.2.4 Predictive Labels And Neighbors with Embeddings Transductively Or Inductively from Data (Planetoid)

On the networks attached with vertex attributes, Planetoid [40] leverages network embedding together with vertex attributes to carry out semi-supervised learning. Planetoid learns vertex embeddings by minimizing the loss for predicting structural context, which is formulated as

$$\mathcal{L}_u = -\mathbb{E}_{(i,c,\gamma)} \log \sigma(\gamma \mathrm{w}_c^{\mathrm{T}} \mathbf{e}_i), \tag{34}$$

where (i, c) is the index for vertex context pair (v_i, v_c), $\mathbf{e}_i$ is the embedding of vertex v_i, w_c is the parameter vector for context vertex v_c, and $\gamma \in \{+1, -1\}$ indicates whether the sampled vertex context pair (i, c) is positive or negative. The triple (i, c, γ) is sampled according to both the network structure and vertex labels, which encodes the information from these two sources into vertex representation $\mathbf{e}_i$.

Planetoid then maps the learned vertex representations $\mathbf{e}$ and vertex attributes $\mathbf{x}$ to hidden layer space via deep neural network, and concatenates these two hidden layer representations together to predict vertex labels, by minimizing the following classification loss:

$$\mathcal{L}_s = -\frac{1}{L} \sum_{i=1}^{L} \log p(y_i | \mathbf{x}_i, \mathbf{e}_i), \tag{35}$$

where

$$p(y_i | \mathrm{x}_i, \mathrm{e}_i) = \frac{\exp\left(\left[\mathrm{h}^k(\mathbf{x}_i)^{\mathrm{T}}, \mathrm{h}^l(\mathbf{e}_i)^{\mathrm{T}}\right] \cdot \mathrm{w}_{y_i}\right)}{\sum_{y'} \exp\left(\left[\mathrm{h}^k(\mathbf{x}_i)^{\mathrm{T}}, \mathrm{h}^l(\mathbf{e}_i)^{\mathrm{T}}\right] \cdot \mathrm{w}_{y'}\right)}. \tag{36}$$

Here, $\mathrm{h}^k(\cdot)$ and $\mathrm{h}^l(\cdot)$ is the deep neural network transformation for vertex attributes $\mathbf{x}$ and vertex embedding $\mathbf{e}$, and w_y is the parameter vector for class y.

To integrate network structure, vertex attributes and vertex labels together, Planetoid jointly minimizes the two objectives (34) and (35). Planetoid is then extended to inductive setting via constructing vertex embedding $\mathbf{e}$ from vertex attributes $\mathbf{x}$ with deep neural network.

5.2.5 Label informed Attribute Network Embedding (LANE)

LANE [26] learns vertex representations by embedding the network structure proximity, attribute affinity, and label proximity into a unified latent representation. The learned representations are expected to capture both the network structure and vertex attribute information, as well as respect the provided label information. The embedding learning in LANE is carried out in two stages. In the first stage, vertex proximity in network structure and attribute information is mapped into latent representations $\mathbf{U}^{(G)}$ and $\mathbf{U}^{(A)}$, then $\mathbf{U}^{(A)}$ is incorporated into $\mathbf{U}^{(G)}$ by maximizing their correlations. In the second stage, LANE employs the joint proximity (determined by $\mathbf{U}^{(G)}$) to smooth label information and uniformly embeds them into another latent representation $\mathbf{U}^{(Y)}$, and then embeds $\mathbf{U}^{(A)}$, $\mathbf{U}^{(G)}$ and $\mathbf{U}^{(Y)}$ into a unified embedding representation $\mathbf{H}$.

Summary: The above semi-supervised content augmented NRL algorithms use three strategies to fully integrate network structure, vertex content and vertex labels to seek more informative and discriminative vertex representations. The first, used by TriDNR [38] and LDE [39]is to maximize the conditional co-occurrence probability of vertex-word relation, vertex-vertex structure relation and vertex-label relation. The second used by DMF [8] and Planetoid [40] is to enforce the content augmented vertex representations with a classification objective. The third used by LANE [26] is to unify vertex structure latent representations, vertex content latent representations and vertex label latent representations via embedding them into a joint space.

6 Applications of Network Representation Learning

Once new vertex representations are learned via network representation learning techniques, traditional vector-based algorithms can be used to solve many important analytic tasks, such as vertex classification, link prediction, clustering, visualization, and recommendation. The effectiveness of the learned representations can also be validated through assessing their performance on these tasks.

6.1 Vertex Classification

Vertex classification is one of the most important tasks among network analytic research. Often in networks, vertices are associated with semantic labels characterizing certain aspects of entities, such as demographics, beliefs, interests, or affiliations. In citation networks, a publication may be labeled with topics or research areas, while the labels of entities in social network may indicate individuals' interests or political beliefs. Because network vertices are partially or sparsely labeled due to high labeling costs, a large portion of vertices in networks have unknown labels. The problem of vertex classification aims to predict the labels of unlabeled vertices given a partially labeled network [10], [11]. Since vertices are not independent of each other but connected in the form of a network via links, vertex classification should exploit these dependencies for jointly classifying the labels of vertices. Among others, collective classification proposes to construct a new set of vertex features that summarize label dependencies in the neighborhood, which has been demonstrated to be most effective in classifying many real-world networks [56], [57].

Network representation learning follows the same principle that automatically learns vertex features based on network structure. Existing studies have evaluated the discriminative power of the learned vertex representations under two settings: unsupervised settings (e.g., [1], [6], [7], [18], [30]), where vertex representations are learned separately, followed by applying discriminative classifiers like SVM or logistic regression on the new embeddings, and semi-supervised settings (e.g., [8], [26], [34], [35], [36]), where representation learning and discriminative learning are simultaneously tackled, so that discriminative power inferred from labeled vertices can directly benefit the learning of informative vertex representations. These studies have proved that better vertex representations can contribute to high classification accuracy.

6.2 Link Prediction

Another important application of network representation learning is link prediction [13], [58], which aims to infer the existence of new relationships or still unknown interactions between pairs of entities based on the currently observed links and their properties. The approaches developed to solve this problem can enable the discovery of implicit or missing interactions in the network, the identification of spurious links, as well as understanding the network evolution mechanism. Link prediction techniques are widely applied in social networks to predict unknown connections among people, which can be used to recommend friendship or identify suspicious relationships. Most of the current social networking systems are using link prediction to automatically suggest friends with a high degree of accuracy. In biological networks, link prediction methods have been developed to predict previously unknown interactions between proteins, thus significantly reducing the costs of empirical approaches. Readers can refer to the survey papers [12], [59] for the recent progress in this field.

Good network representations should be able to capture explicit and implicit connections between network vertices thus enabling application to link prediction. [17] and [31] predict missing links based on the learned vertex representations on social networks. [30] also applies network representation learning to collaboration networks and protein-protein interaction networks. They demonstrate that on these networks links predicted using the learned representations achieve better performance than traditional similarity-based link prediction approaches.

6.3 Clustering

Network clustering refers to the task of partitioning a network into a set of clusters, such that vertices are densely connected to each other within the same cluster, but connected to few vertices from other clusters [60]. Such cluster structures, or communities widely occur in a wide spectrum of networked systems from bioinformatics, computer science, physics, sociology, etc., and have strong implications. For example, in biology networks, clusters may correspond to a group of proteins having the same specific function; in the network of Webpages, clusters are likely pages having similar topics or related content; in social networks, clusters may indicate groups of people having similar interests or affiliations.

Researchers have proposed a large body of network clustering algorithms based on various metrics of similarity or strength of connection between vertices. Min-max cut and normalized cut methods [61], [62] seek to recursively partition a graph into two clusters that maximize the number of intra-cluster connections and minimize the number of inter-cluster connections. Modularity-based methods (e.g., [63], [64]) aim to maximize the modularity of a clustering, which is the fraction of intra-cluster edges minus the expected fraction assuming the edges were randomly distributed. A network partitioning with high modularity would have dense intra-cluster connections but sparse inter-cluster connections. Some other methods (e.g., [65]) try to identify nodes with similar structural roles like bridges and outliers.

Recent NRL methods (i.e., GraRep [22], DNGR [9], M-NMF [24], and pRBM [25]) used the clustering performance to evaluate the quality of the learned network representations on different networks. Intuitively, better representations would lead to better clustering performance. These works followed the common approach that first applies an unsupervised NRL algorithm to learn vertex representations, and then performs k-means clustering on the learned representations to cluster the vertices. In particular, pRBM [25] showed that the performance of NRL methods outperforms the baseline that uses original features for clustering without learning representations. This suggests that effective representation learning can improve the clustering performance.

6.4 Visualization

Visualization techniques play critical roles in managing, exploring, and analyzing complex networked data. [66] surveys a range of methods used to visualize graphs from an information visualization perspective. This work compares various traditional layouts used to visualize graphs, such as tree-, 3D-, and hyperbolic-based methods, and shows that classical visualization techniques are proved effective for small or intermediate sized networks; they however confront a big challenge when applied to large-scale networks. Few systems can claim to deal effectively with thousands of vertices, although networks with this order of magnitude often occur in a wide variety of applications. Consequently, a first step in the visualization process is often to reduce the size of the network to display. One common approach is essentially to find an extremely low-dimensional representation of a network that preserves the intrinsic structure, i.e., keeping similar vertices close and dissimilar vertices far apart, in the low-dimensional space [14].

Network representation learning has the same objective that embeds a large network into a new latent space of low dimensionality. After new embeddings are obtained in the vector space, popular methods such as t-distributed stochastic neighbor embedding (t-SNE) [67] can be applied to visualize the network in a 2-D or 3-D space. By taking the learned vertex representations as input, LINE [1] used the t-SNE package to visualize the DBLP co-author network after the authors are mapped into a 2-D space, and showed that LINE is able to cluster authors in the same field to the same community. HSCA [18] illustrated the advantages of the content-augmented NRL algorithm by visualizing the citations networks. Semi-supervised algorithms (e.g., TLINE [36], TriDNR [38], and DMF [8]) demonstrated that the visualization results have better clustering structures with vertex labels properly imported.

6.5 Recommendation

In addition to structure, content, and vertex label information, many social networks also include geographical and spatial-temporal information, and users can share their experiences online with their friends for point of interest (POI) recommendation, e.g., transportation, restaurant, and sightseeing landmark, etc. Examples of such location-based social networks (LBSN) include Foursquare, Yelp, and Facebook Places, and many others. For these types of social

networks, POI recommendation intends to recommend user interested objects, depending on their own context, such as the geographic location of the users and their interests. Traditionally, this is solved using approaches such as collaborative filtering, to leverage spatial and temporal correlation between user activities and geographical distance [68]. However, because each user's check-in records are very sparse, finding similar users or calculating transition probability between users and locations is a significant challenge.

Recently, spatial-temporal embedding [15] has emerged to learn low-dimensional dense vectors to represent users, locations, and point-of-interests etc. As a result, each user, each location, and each POI can be represented as a low-dimensional vector, respectively, for similarity search and many other analysis. An inherent advantage of such spatial-temporal aware embedding is that it alleviates the data sparsity problem, because the learned low dimensional vector is typically much more dense than the original representation, it makes query tasks, such as top-k POI search, much more accurate than traditional approaches.

6.6 Knowledge Graph

Knowledge graphs represent a new type of data structure in database systems which encode structured information of billions of entities and their rich relations. A knowledge graph typically contains a rich set of heterogeneous objects and different types of entity relationships. Such networked entities form a gigantic graph and is now powering many commercial search engines to find similar objects online. Traditionally, knowledge graph search is carried out through database driven approaches to explore schema mapping between entities, including entity relationships. Recent advancement in network representation learning has proposed new solutions to learn structured embeddings of knowledge bases [69]. Such embedding learning methods intend to learn a low-dimensional vector representation for knowledge graph entities, such that generic database queries, such as top-k search, can be carried out by comparing vector representation of the query object and objects in the database.

In addition to using vector representation to represent knowledge graph entities, research has also proposed to use such representation to further enhance and complete the knowledge graph itself. For example, knowledge graph completion intends to discover complete relationships between entities, and a recent work [70] has proposed to use graph context to find missing links between entities. This is similar to link prediction in social networks, but the entities are typically heterogeneous and a pair of entities may also have different types of relationships.

7 Evaluation for Network Representation Learning

In order to make fair comparisons of the effectiveness of different NRL algorithms, a broad range of real-word information networks have been used in the literature for analysis and evaluation. In this section, we summarize commonly used benchmark datasets and the network analytic tasks that are adopted as evaluation methods.

7.1 Benchmark Datasets

Network representation learning techniques are developed and evaluated based on real-world networks. Benchmark datasets play an important role for the research community to evaluate the performance of newly developed NRL algorithms as compared to the existing baseline methods. So far, several network datasets have been made publicly available to facilitate the evaluation of NRL algorithms across different tasks. We summarize a list of network datasets used by most of the published network representation learning papers in Table 4.

Table 4 summarizes the main characteristics of the publicly available benchmark datasets, including the type of network (directed or undirected, binary or weighted), number of vertices $|V|$, number of edges $|E|$, number of labels $|\mathcal{Y}|$, whether the network is multi-labeled or not, as well as whether network vertices are attached with attributes. In Table 4, according to the property of information networks, we classify benchmark datasets into six different types:

1) **Social Network**. The BlogCatalog, Flickr and YouTube datasets are formed by the users of the corresponding online social network platforms. For the three datasets, vertex labels are defined by user interest groups and user attributes are not collected. The Amherst, Hamilton, Mich and Rochester [71] datasets are the Facebook networks that are formed by users from respective US universities, where each user is associated with six user profile features. Different from rich text features, the user profile features in social networks are usually topic-inconsistent and long-tail distributed.
2) **Language Network**. The language network Wikipedia [1] is a word co-occurrence network constructed from the entire set of English Wikipedia pages. There is no class label for the Wikipedia network. The word embeddings learned from this language network is evaluated by word analogy and document classification.
3) **Citation Network**. The citation networks are directed information networks formed by author-author citation relationships or paper-paper citation relationships. They are collected from different databases of academic papers, such as DBLP and Citeseer. Among the commonly used citation networks, DBLP (AuthorCitation) [1] is a weighted citation network between authors with the edge weight defined by the number of papers written by one author and cited by the other author, while DBLP (PaperCitation) [1], Cora, Citeseer, PubMed and Citeseer-M10 are the binary paper citation networks, which are also attached with vertex text attributes as the content of papers. Compared with user profile features in social network, the vertex text features here are more topic-centric, informative and can better complement network structure to learn more informative vertex representations.
4) **Collaboration Network**. The collaboration network Arxiv GR-QC [73] describes the co-author relationships for papers in the research field of General Relativity and Quantum Cosmology. In this network,

TABLE 4
A summary of benchmark datasets for evaluating network representation learning.

Category	Dataset	Type	$\|V\|$	$\|E\|$	$\|\mathcal{Y}\|$	Multi-label	Vertex attr.
Social Network	BlogCatalog[a]	undirected, binary	10,312	333,983	39	Yes	No
	Flickr[b]	undirected, binary	80,513	5,899,882	195	Yes	No
	YouTube[b]	undirected, binary	1,138,499	2,990,443	47	Yes	No
	Amherst[c] [71]	undirected, binary	2,021	81,492	15	Yes	Yes
	Hamilton[c] [71]	undirected, binary	2,118	87,486	15	Yes	Yes
	Mich[c] [71]	undirected, binary	2,933	54,903	13	Yes	Yes
	Rochester[c] [71]	undirected, binary	4,145	145,305	19	Yes	Yes
Language Network	Wikipedia [1]	undirected, weighted	1,985,098	1,000,924,086	N/A	N/A	No
Citation Network	DBLP (PaperCitation) [1], [72]	directed, binary	781,109	4,191,677	7	No	Yes
	DBLP (AuthorCitation) [1], [72]	directed, weighted	524,061	20,580,238	7	No	No
	Cora[d]	directed, binary	2,708	5,429	7	No	Yes
	Citeseer[d]	directed, binary	3,312	4,732	6	No	Yes
	PubMed[d]	directed, binary	19,717	44,338	3	No	Yes
	Citeseer-M10[e]	directed, binary	10,310	77,218	10	No	Yes
Collaboration network	Arxiv GR-QC [73]	undirected, binary	5,242	28,980	N/A	N/A	No
Webpage Network	Wikipedia[d]	directed, binary	2,405	17,981	20	No	Yes
	WebKB[d]	directed, binary	877	1,608	5	No	Yes
	Political Blog [74]	directed, binary	1,222	16,715	2	No	No
Biological Network	Protein-Protein Interaction [75]	undirected, binary	4,777	184,812	40	Yes	No

[a]http://www.public.asu.edu/~ltang9/
[b]http://socialnetworks.mpi-sws.org/data-imc2007.html
[c]https://escience.rpi.edu/data/DA/fb100/
[d]http://linqs.cs.umd.edu/projects//projects/lbc
[e]http://citeseerx.ist.psu.edu/

vertices represent authors and edges indicate co-author relationships between authors. Because there is no category information for vertices, this network is used for the link prediction task to evaluate the quality of learned vertex representations.

5) **Webpage Network**. Webpage Networks (Wikipedia, WebKB and Political Blog [74]) are composed of real-world webpapges and the hyperlinks between them. In these networks, the vertex represent a webpage and the edge indicates that there is a hyperlink from one webpage to another. For webpage networks, the webpage text content is also collected to serve as vertex features.
6) **Biological Network**. The Protein-Protein Interaction network [75] is a subgraph of the PPI network for Homo Sapiens. In this biological network, the vertex represents a protein and the edge indicates that there is an interaction between proteins. The labels of vertices are obtained from the hallmark gene sets [76] and represent biological states.

7.2 Evaluation Methods

The quality of the vertex representations learned by different NRL algorithms cannot be directly compared because of the unavailability of ground truth information. Alternatively, in order to evaluate the effectiveness of NRL algorithms on learned vertex representations, several network analytic tasks are commonly used for comparison studies:

1) **Network Reconstruction**. The task of network reconstruction is to reconstruct the original network from the learned vertex representations by predicting the links between vertices based on the inner product or similarity between vertex representations. The known links in the original network are served as the ground-truth for evaluating reconstruction performance. $precision@k$ and MAP [17] are often used as the evaluation metrics. This evaluation method can check whether the learned vertex representations well preserve network structure and support the formation of the network.
2) **Vertex Classification**. As an evaluation method for NRL, vertex classification is conducted by taking learned vertex representations as features. After a classifier is trained on labeled vertices using these features, it is used to classify unlabeled vertices. The classification performance is used to evaluate the quality of the learned vertex representations. Different settings of vertex classification, including binary-class classification, multi-class classification and multi-label classification, are often carried out, depending on the underlying network characteristics. For binary-class classification, F_1 score is used as the evaluation criterion. For multi-class and multi-label classification, $Micro\text{-}F_1$ and $Micro\text{-}F_1$ are adopted as evaluation criteria.
3) **Vertex Clustering**. To validate the effectiveness of NRL algorithms, vertex clustering is also carried out by applying the k-means algorithm to the learned vertex representations. Communities in networks are served as the ground truth to assess the quality of clustering results, which is measured by

TABLE 5
A summary of NRL algorithms with respect to the evaluation methodology

Category	Algorithm	Network Type	Evaluation Method	Code Link
Unsupervised	Social Dim. [27], [28], [29]	Social Network	Vertex Classification	
	DeepWalk [6]	Social Network	Vertex Classification	https://github.com/phanein/deepwalk
	LINE [1]	Citation Network Language Network Social Network	Vertex Classification Visualization	https://github.com/tangjianpku/LINE
	GraRep [22]	Citation Network Language Network Social Network	Vertex Classification Vertex Clustering Visualization	https://github.com/ShelsonCao/GraRep
	DNGR [9]	Language Network	Vertex Clustering Visualization	https://github.com/ShelsonCao/DNGR
	SDNE [17]	Collaboration Network Language Network Social Network	Network Reconstruction Vertex Classification Link Prediction Visualization	https://github.com/suanrong/SDNE
	node2vec [30]	Biological Network Language Network Social Network	Vertex Classification Link Prediction	https://github.com/aditya-grover/node2vec
	HOPE [31]	Social Network Citation Network	Network Reconstruction Link Prediction	
	APP [32]	Social Network Citation Network Collaboration Network	Link Prediction	
	M-NMF [24]	Social Network Webpage Network	Vertex Classification Vertex Clustering	
	TADW [7]	Citation Network Webpage Network	Vertex Classification	https://github.com/thunlp/tadw
	HSCA [18]	Citation Network Webpage Network	Vertex Classification Visualization	https://github.com/daokunzhang/HSCA
	pRBM [25]	Social Network	Vertex Clustering	
	UPP-SNE [33]	Social Network	Vertex Classification Vertex Clustering	
Semi-supervised	DDRW [34]	Social Network	Vertex Classification	
	MMDW [35]	Citation Network Webpage Network	Vertex Classification Visualization	https://github.com/thunlp/MMDW
	TLINE [36]	Citation Network Collaboration Network	Vertex Classification Visualization	
	GENE [37]	Social Network	Vertex Classification	
	TriDNR [38]	Citation Network	Vertex Classification Visualization	https://github.com/shiruipan/TriDNR
	LDE [39]	Social Network Citation Network	Vertex Classification	
	DMF [8]	Citation Network	Vertex Classification Visualization	https://github.com/daokunzhang/DMF_CC
	Planetoid [40]	Citation Network	Vertex Classification Visualization	https://github.com/kimiyoung/planetoid
	LANE [26]	Social Network	Vertex Classification	

$Accuracy$ and NMI (normalized mutual information) [77]. The hypothesis is that, if the learned vertex representations are indeed informative, vertex clustering on learned vertex representations should be able to discover community structures. In other words, good vertex representations are expected to generate good vertex clustering performance.

4) **Link Prediction**. Link prediction based on learned vertex representations can be used to evaluate whether the learned vertex representations are informative enough to support the network evolution mechanism. In order to carry out link prediction on a given information network, using learned vertex representations, a portion of edges are firstly removed. After that, vertex representations are learned from the remaining network. Finally, the removed edges are predicted with learned vertex representations. The performance of link prediction is measured by AUC and $precision@k$.

5) **Visualization**. Visualization on learned vertex representations provides a straightforward way to evaluate learned vertex representations with human eyes. After vertex representations are learned, t-distributed stochastic neighbor embedding (t-SNE) [67] is applied to project vertex representation vectors into a 2-D space. The distribution of vertex 2-D mappings in the 2-D space can be easily visualized. If the learned vertex representations are of good quality, in the 2-D space, vertices within a same class or community are expected to embedded

closely and the 2-D mappings of vertices in different classes or communities are expected to be far away from each other.

In Table 5, we summarize the type of information networks and network analytic tasks used to evaluate the quality of the learned representations of existing NRL algorithms. We also provide links for the codes of respective NRL algorithms if available to help interested readers to further study these algorithms or run experiments for comparison. Overall, Table 5 shows that social networks and citation networks are frequently used as benchmark datasets and vertex classification is most commonly used as the evaluation method in both unsupervised and semi-supervised settings.

8 Conclusions and Future Research Directions

The availability of large-scale information networks has encouraged increasing attention on deriving useful network representations to facilitate various network analytic tasks. This survey provides a comprehensive review of the state-of-the-art network representation learning algorithms that are proposed in the current data mining and machine learning literature. We first introduced the problem definition, network properties preserved by various representation learning methods, as well as the challenges faced by network representation learning in general. Then, we proposed a taxonomy (Fig. 2) to summarize existing techniques into two settings: unsupervised setting and semi-supervised settings. In both settings, our taxonomy further categorizes methods into subgroups, depending on the information sources they use (Table 2) and the methodologies they employ (Table 3), and reviewed representative algorithms for each subgroup. We also summarized the benchmark datasets (Table 4) and the evaluation methods (Table 5) used for validating existing NRL algorithms. We believe that our taxonomy and review not only help researchers to gain a comprehensive understanding of methods in the field, understand the strength and uniqueness of different approaches, but also provide rich resources to advance the research in network representation learning.

Non-linearity. Earlier research has primarily focused on preserving various types of structure information and demonstrated great promise in various tasks. More recent studies have devoted to leveraging the information from vertex attributes and/or vertex labels to enhance network representation learning. From the methodology perspective, the majority of the existing techniques are matrix factorization and random walk based, while deep learning based methods have explored the use of non-linear models that have the ability to capture more complex network structure. Yet, finding effective ways to improve the interpretability of deep learning based methods remains an open research problem.

Task-dependence. To date, most existing NRL algorithms are task-independent, and task-specific NRL algorithms have primarily focused on vertex classification under the semi-supervised setting. However, network representation learning tailored to many other important network analysis tasks like link prediction [12], community detection [78], and anomaly detection [79], has not been extensively investigated. A good task-specific NRL algorithm must preserve information critical to the specific task in order to optimize the performance. For example, semi-supervised NRL algorithms leverage vertex labels to learn discriminative vertex representations for maximizing vertex classification performance. Algorithms like HOPE [31] and APP [49] learn effective vertex representations for link prediction by preserving the asymmetric structural proximity. At the core of task-specific network representation learning is the ability to identify and preserve the desired network properties that can best benefit the target task.

Network dynamics. Current research on network representation learning has mainly concerned static networks. However, in reality, networks always evolve over time, i.e., vertices or edges are constantly added or deleted. Therefore, network representation learning is expected to work in an online mode to account for new added vertices, i.e., predict the representations of new added vertices. This is referred to as "out-of-sample" problem. A few studies such as [40], [80] have attempted to exploit inductive learning to address this issue. They focus on learning an explicit mapping function from an existing network at a snapshot, and using this function to infer the representations of out-of-sample vertices, based on their available information such as attributes or neighborhood structure. However, this line of research has not considered the effect of out-of-sample vertices on the mapping function. How to adapt the static network representation learning to better handle network dynamics still needs further exploration.

Scalability. The scalability is another driving factor to advance the research on network representation learning. Several NRL algorithms have made attempts to scale to large-scale networks with linear time complexity with respect to the number of vertices/edges. Nevertheless, the scalability still remains a major challenge for network representation learning. In analyzing and comparing the time complexity of various algorithms, we notice that random walk, edge modeling, and deep learning based methods that adopt stochastic gradient descent optimization are much more efficient than matrix factorization based methods that are solved by eigen decomposition and alternative optimization. Given that matrix factorization based methods have shown great promise in incorporating vertex content attributes and discovering community structures, efficient solutions like on-line matrix factorization or parallel matrix factorization need to be investigated in order to scale to networks with billions of vertices. The high dimensionality of vertex attributes also increase the difficulty for designing efficient NRL algorithms.

Heterogeneity. All studies reviewed in this survey focus on network representation learning problems in homogeneous networks, where all vertices are of the same type. However, as the amount of multimedia data (i.e., images, videos) rapidly increases, there is an increasing need to study heterogeneous networks with different types of vertices and links, such as DBLP, DBpedia, and Flickr. Heterogeneous networks are composed of different types of entities, such as text, images, or videos, and the relationships between entities are much more complex. This makes it even

more difficult to measure the proximity between vertices and seek a common and coherent embedding space. Very recent studies by [81], [82], [83] have started to investigate heterogeneous network embedding. However, the research along this line is still at very early stage. Further research requires to investigate better ways for capturing the proximity between cross-modal data, and in particular their interplay with network structure.

REFERENCES

[1] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-scale information network embedding," in *Proceedings of the 24th International Conference on World Wide Web*. ACM, 2015, pp. 1067–1077.

[2] F. D. Malliaros and M. Vazirgiannis, "Clustering and community detection in directed networks: A survey," *Physics Reports*, vol. 533, no. 4, pp. 95–142, 2013.

[3] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[4] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[5] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Advances in Neural Information Processing Systems*, 2002, pp. 585–591.

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

[7] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, "Network representation learning with rich text information," in *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, 2015, pp. 2111–2117.

[8] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Collective classification via discriminative matrix factorization on sparsely labeled networks," in *Proceedings of the 25th ACM International Conference on Information and Knowledge Management*. ACM, 2016, pp. 1563–1572.

[9] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning graph representations," in *Proceedings of the 30th AAAI Conference on Artificial Intelligence*. AAAI Press, 2016, pp. 1145–1152.

[10] S. Zhu, K. Yu, Y. Chi, and Y. Gong, "Combining content and link for classification using matrix factorization," in *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2007, pp. 487–494.

[11] S. Bhagat, G. Cormode, and S. Muthukrishnan, "Node classification in social networks," *Social Network Data Analytics*, pp. 115–148, 2011.

[12] L. Lü and T. Zhou, "Link prediction in complex networks: A survey," *Physica A: Statistical Mechanics and its Applications*, vol. 390, no. 6, pp. 1150–1170, 2011.

[13] S. Gao, L. Denoyer, and P. Gallinari, "Temporal link prediction by integrating content and structure information," in *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*. ACM, 2011, pp. 1169–1174.

[14] J. Tang, J. Liu, and Q. Mei, "Visualizing large-scale and high-dimensional data," in *Proceedings of the 25th International Conference on World Wide Web*. ACM, 2016, pp. 287–297.

[15] M. Xie, H. Yin, H. Wang, F. Xu, W. Chen, and S. Wang, "Learning graph-based poi embedding for location-based recommendation," in *Proceedings of the 25th ACM International Conference on Information and Knowledge Management*. ACM, 2016, pp. 15–24.

[16] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *Proceedings of the 29th AAAI Conference on Artificial Intelligence*. AAAI, 2015, pp. 2181–2187.

[17] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 1225–1234.

[18] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Homophily, structure, and content augmented network representation learning," in *Proceedings of the 16th IEEE International Conference on Data Mining*. IEEE, 2016, pp. 609–618.

[19] L. G. Moyano, "Learning network representations," *The European Physical Journal Special Topics*, vol. 226, no. 3, pp. 499–518, 2017.

[20] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *arXiv:1705.02801*, 2017.

[21] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *To appear in the IEEE Data Engineering Bulletin, arXiv:1709.05584*, 2017.

[22] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in *Proceedings of the 24th ACM International Conference on Information and Knowledge Management*. ACM, 2015, pp. 891–900.

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

[24] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding," in *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, 2017, pp. 203–209.

[25] S. Wang, J. Tang, F. Morstatter, and H. Liu, "Paired restricted boltzmann machine for linked data," in *Proceedings of the 25th ACM International Conference on Information and Knowledge Management*. ACM, 2016, pp. 1753–1762.

[26] X. Huang, J. Li, and X. Hu, "Label informed attributed network embedding," in *Proceedings of the 10th ACM International Conference on Web Search and Data Mining*. ACM, 2017, pp. 731–739.

[27] L. Tang and H. Liu, "Relational learning via latent social dimensions," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2009, pp. 817–826.

[28] ——, "Leveraging social media networks for classification," *Data Mining and Knowledge Discovery*, vol. 23, no. 3, pp. 447–478, 2011.

[29] ——, "Scalable learning of collective behavior based on sparse social dimensions," in *Proceedings of the 18th ACM International Conference on Information and Knowledge Management*. ACM, 2009, pp. 1107–1116.

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

[31] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 1105–1114.

[32] C. Zhou, Y. Liu, X. Liu, Z. Liu, and J. Gao, "Scalable graph embedding for asymmetric proximity," in *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, 2017, pp. 2942–2948.

[33] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "User profile preserving social network embedding," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 2017, pp. 3378–3384.

[34] J. Li, J. Zhu, and B. Zhang, "Discriminative deep random walk for network classification," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, vol. 1, 2016, pp. 1004–1013.

[35] C. Tu, W. Zhang, Z. Liu, and M. Sun, "Max-margin deepwalk: discriminative learning of network representation," in *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, 2016, pp. 3889–3895.

[36] X. Zhang, W. Chen, and H. Yan, "Tline: scalable transductive network embedding," in *Information Retrieval Technology*. Springer, 2016, pp. 98–110.

[37] J. Chen, Q. Zhang, and X. Huang, "Incorporate group information to enhance network embedding," in *Proceedings of the 25th ACM International Conference on Information and Knowledge Management*. ACM, 2016, pp. 1901–1904.

[38] S. Pan, J. Wu, X. Zhu, C. Zhang, and Y. Wang, "Tri-party deep network representation," in *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, 2016, pp. 1895–1901.

[39] S. Wang, J. Tang, C. Aggarwal, and H. Liu, "Linked document embedding for classification," in *Proceedings of the 25th ACM International Conference on Information and Knowledge Management*. ACM, 2016, pp. 115–124.

[40] Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," in *Proceedings of the*

33rd International Conference on International Conference on Machine Learning (ICML), 2016, pp. 40–48.

[41] M. E. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Physical review E*, vol. 74, no. 3, p. 036104, 2006.

[42] N. Natarajan and I. S. Dhillon, "Inductive matrix completion for predicting gene–disease associations," *Bioinformatics*, vol. 30, no. 12, pp. i60–i68, 2014.

[43] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[44] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, 2013, pp. 3111–3119.

[45] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[46] R. Salakhutdinov and G. Hinton, "Semantic hashing," *International Journal of Approximate Reasoning*, vol. 50, no. 7, pp. 969–978, 2009.

[47] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[48] L. Katz, "A new status index derived from sociometric analysis," *Psychometrika*, vol. 18, no. 1, pp. 39–43, 1953.

[49] H. H. Song, T. W. Cho, V. Dave, Y. Zhang, and L. Qiu, "Scalable proximity estimation and link prediction in online social networks," in *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference*. ACM, 2009, pp. 322–335.

[50] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Advances in neural information processing systems*, 2008, pp. 1177–1184.

[51] J. Zhu, A. Ahmed, and E. P. Xing, "Medlda: maximum margin supervised topic models," *Journal of Machine Learning Research*, vol. 13, no. Aug, pp. 2237–2278, 2012.

[52] J. Mairal, J. Ponce, G. Sapiro, A. Zisserman, and F. R. Bach, "Supervised dictionary learning," in *Advances in Neural Information Processing Systems*, 2009, pp. 1033–1040.

[53] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *Journal of Machine Learning Research*, vol. 9, no. Aug, pp. 1871–1874, 2008.

[54] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on Machine Learning*, 2014, pp. 1188–1196.

[55] N. Djuric, H. Wu, V. Radosavljevic, M. Grbovic, and N. Bhamidipati, "Hierarchical neural language models for joint representation of streaming documents and their content," in *Proceedings of the 24th International Conference on World Wide Web*. ACM, 2015, pp. 248–255.

[56] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI Magazine*, vol. 29, no. 3, p. 93, 2008.

[57] P. Kazienko and T. Kajdanowicz, "Label-dependent node classification in the network," *Neurocomputing*, vol. 75, no. 1, pp. 199–209, 2012.

[58] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *Journal of the American Society for Information Science and Technology*, vol. 58, no. 7, pp. 1019–1031, 2007.

[59] V. Martinez, F. Berzal, and J.-C. Cubero, "A survey of link prediction in complex networks," *ACM Computing Surveys*, vol. 49, no. 4, 2017.

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

[61] C. H. Q. Ding, X. He, H. Zha, M. Gu, and H. D. Simon, "A min-max cut algorithm for graph partitioning and data clustering," in *Proceedings of the 2001 IEEE International Conference on Data Mining*. IEEE, 2001, pp. 107–114.

[62] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[63] N. M. E. J. and M. Girvan, "Finding and evaluating community structure in networks," *Physics Review*, vol. 69, no. 026113, 2004.

[64] M. E. Newman, "Modularity and community structure in networks," *Proceedings of the National Academy of Sciences*, vol. 103, no. 23, pp. 8577–8582, 2006.

[65] X. Xu, N. Yuruk, Z. Feng, and T. A. J. Schweiger, "SCAN: A structural clustering algorithm for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2007, pp. 824–833.

[66] I. Herman, G. Melançon, and M. S. Marshall, "Graph visualization and navigation in information visualization: A survey," *IEEE Transactions on Visualization and Computer Graphics*, vol. 6, no. 1, pp. 24–43, 2000.

[67] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[68] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma, "Mining interesting locations and travel sequences from gps trajectories," in *Proceedings of the World Wide Web International Conference*, 2009, pp. 791–800.

[69] A. Bordes, J. Weston, R. Collobert, and Y. Bengio, "Mining interesting locations and travel sequences from gps trajectories," in *Proceedings of the 25th AAAI Conference on Artificial Intelligence*, 2011, pp. 301–306.

[70] J. Feng, M. Huang, Y. Yang *et al.*, "Gake: Graph aware knowledge embedding," in *Proceedings of the 26th International Conference on Computational Linguistics (COLING)*, 2016, pp. 641–651.

[71] A. L. Traud, P. J. Mucha, and M. A. Porter, "Social structure of facebook networks," *Physica A: Statistical Mechanics and its Applications*, vol. 391, no. 16, pp. 4165–4180, 2012.

[72] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, "Arnetminer: extraction and mining of academic social networks," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2008, pp. 990–998.

[73] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph evolution: densification and shrinking diameters," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, p. 2, 2007.

[74] L. A. Adamic and N. Glance, "The political blogosphere and the 2004 us election: divided they blog," in *Proceedings of the 3rd International Workshop on Link Discovery*. ACM, 2005, pp. 36–43.

[75] B.-J. Breitkreutz, C. Stark, T. Reguly, L. Boucher, A. Breitkreutz, M. Livstone, R. Oughtred, D. H. Lackner, J. Bähler, V. Wood *et al.*, "The biogrid interaction database: 2008 update," *Nucleic Acids Research*, vol. 36, pp. D637–D640, 2008.

[76] A. Liberzon, A. Subramanian, R. Pinchback, H. Thorvaldsdóttir, P. Tamayo, and J. P. Mesirov, "Molecular signatures database (msigdb) 3.0," *Bioinformatics*, vol. 27, no. 12, pp. 1739–1740, 2011.

[77] A. Strehl and J. Ghosh, "Cluster ensembles – a knowledge reuse framework for combining multiple partitions," *Journal of Machine Learning Research*, vol. 3, pp. 583–617, 2003.

[78] J. Yang, J. McAuley, and J. Leskovec, "Community detection in networks with node attributes," in *Proceedings of the 2013 IEEE International Conference on Data Mining*. IEEE, 2013, pp. 1151–1156.

[79] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: methods, systems and tools," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 303–336, 2014.

[80] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *arXiv preprint arXiv:1706.02216*, 2017.

[81] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, and T. S. Huang, "Heterogeneous network embedding via deep architectures," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 119–128.

[82] Z. Huang and N. Mamoulis, "Heterogeneous information network embedding for meta path based proximity," in *arXiv:1701.05291*, 2017.

[83] Y. Dong, N. V. Chawla, and A. Swami, "Metapath2vec: Scalable representation learning for heterogeneous networks," in *Proceedings of the 23th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 135–144.