



# Network representation learning: models, methods and applications

Anuraj Mohan<sup>1</sup>  · K. V. Pramod<sup>2</sup>

© Springer Nature Switzerland AG 2019

## Abstract

With the rise of large-scale social networks, network mining has become an important sub-domain of data mining. Generating an efficient network representation is one important challenge in applying machine learning to network data. Recently, representation learning methods are widely used in various domains to generate low dimensional latent features from complex high dimensional data. A significant amount of research effort is made in the past few years to generate node representations from graph-structured data using representation learning methods. Here, we provide a detailed study of the latest advancements in the field of network representation learning (also called network embedding). We first discuss the basic concepts and models of network embedding. Further, we build a taxonomy of network embedding methods based on the type of networks and review the major research works that come under each category. We then cover the major datasets used in network embedding research and describe the major applications of network embedding with respect to various network mining tasks. Finally, we provide various directions for future work which enhance further research.

**Keywords** Network embedding · Representation learning · Deep learning · Social networks · Neural networks

**JEL Classification** 06 Computer Science · 02

## 1 Introduction

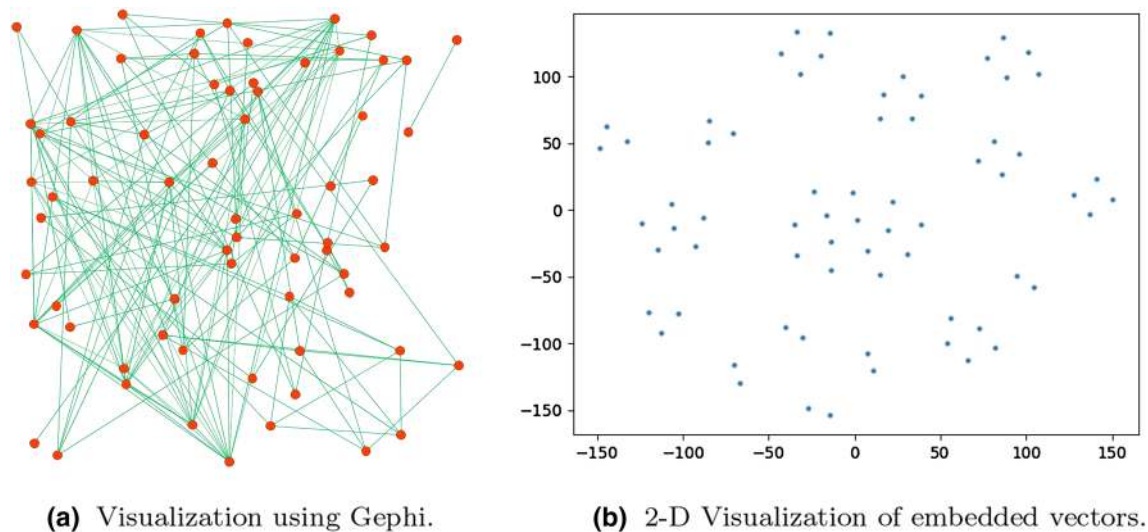
Networks provide a fundamental model for defining a relationship between various entities. Networks have applications in diverse domains like social computing [82], systems biology [76, 141], cyber-physical systems [148], recommender system [74], language modeling [125], and network medicine [4]. Social network reflects the relationship between people, citation network relates research papers, a biological network can define protein–protein interactions, a word co-occurrence network defines linguistic relationships and many more. Analysis and mining of these complex networks can generate various insights, which can be very useful for both scientific and business community. Friend recommendation in social networks, protein function prediction from protein interaction

networks, terrorist group identification from communication networks, influential paper detection from citation networks etc. are some typical examples. We usually define these tasks formally as link prediction [71], node classification [8], graph clustering [105], and influential node detection [79]. Performing these tasks on large real-world networks pose various challenges.

Traditional methods for network embedding use graph algorithm based approaches, which uses adjacency matrix as network representation. Also, these methods adopt iterative processing, which results in high computational cost when applied to large networks. For example, for node classification, most of the approaches like iterative classification algorithm (ICA) [91] and label propagation [149] are iterative approaches. Machine learning methods cannot be directly applied to networks because such

✉ Anuraj Mohan, anurajmohan@gmail.com; K. V. Pramod, pramodkv4@gmail.com | <sup>1</sup>Artificial Intelligence Lab, Department of Computer Applications, Cochin University of Science and Technology, Kerala 682022, India. <sup>2</sup>Department of Computer Applications, Cochin University of Science and Technology, Kerala 682022, India.





**Fig. 1** Subset of github user interaction network

methods assume that the data have independent and identical distribution (i.i.d), which is not true in the case of graph-structured data. Using sparse adjacency matrix representation is also not practical to perform machine learning. An alternate method to perform machine learning on network data is to use hand engineered features generated using network statistics and other measures [34], which is a time-consuming process. Traditional distributed computing platforms [26] are not well suited for parallel processing of graph-structured data. Many specialized distributed graph analytic platforms like Pregel [78], Giraph [81] and Graphx [137] are developed, but their efficiency is limited by the complex phenomenon of real-world networks like scale-free property and power law distributions.

An interesting direction towards applying machine learning on network data is to map the data to a low dimensional latent space, and then to apply traditional machine learning algorithms. This process of mapping the network data to vector space is known as network embedding. Many linear and non-linear dimensionality reduction methods [126] were initially used to generate network embedding. Most of these methods were based on matrix factorization, and hence suffered from scalability issue. More recently, machine learning community has come up with new theories and architectures to learn complex features from high dimensional data. These approaches are referred to as representation learning,<sup>1</sup> which aims at finding a set of transformations that can map the high

dimensional data to a low dimensional manifold. With the success of representation learning on image [44, 60, 128, 134], speech [23, 40, 48], and natural language processing [19, 21, 108], researchers attempted to use these methods on network data and created fruitful results.

Given an input network, we can generate embedding in different output formats, which includes node, edge, subgraph and whole-graph embedding. Edge embedding aims to map the edges of a network to a latent space, and subgraph embedding attempts to map the graph components (subgraph structures) to a vector space. Whole-graph embedding aims to generate the representation of a complete graph in vector space, and many works used graph kernel methods [3, 127] for generating whole-graph representations. Node embedding, which represents vertices of a graph in vector space, is the more focused and well-studied problem, which is covered throughout this survey. Figure 1a shows Gephi [5] visualization of a small subset of github user interaction network, and Fig. 1b shows its 2-D representation in vector space, generated by DeepWalk [96], and plotted using t-SNE [75]. Generating low dimensional vectors as node embedding from a large real-world network is not straightforward. The vector representation should preserve the structural properties of the network which includes the first order, second order and higher order proximities between nodes. The network data is highly sparse and usually non-linear, and the embedding algorithm should generate the embedding from sparse and non-linear data. Many real-world networks contain millions of nodes and edges, and the embedding algorithm should be scalable. In reality, many networks may be heterogeneous, attributed, scale-free

<sup>1</sup> The terms network representation learning and network embedding are used interchangeably in this paper.

and dynamic, and the embedding method should adapt to all such situations.

A few efforts are already made to survey [22, 38, 46, 89] the various approaches for network embedding. In this survey, we focus on the recent methods for node embedding which are inspired by the recent advancements in representation learning. We provide a taxonomy of node embedding methods based on the type of the networks. Networks are classified into broader categories such as homogeneous networks, heterogeneous networks, attributed networks, signed networks, and dynamic networks. We discuss the common models of network representation learning and reviews the major works which come under each model with respect to each type of network. Further, we discuss the applications of network embedding along with the data sets used in the network embedding research.

## 2 Terminologies and problem definition

**Definition 1** A Network is a graph  $G = (V, E)$ , where  $V = \{v_1, v_2 \dots v_n\}$ , is the set of vertices and  $e \in E$  is an edge between any two vertices. An adjacency matrix  $A$  defines the connectivity of  $G$ ,  $A_{ij} = 1$  if  $v_i$  and  $v_j$  are connected, else  $A_{ij} = 0$ .

**Definition 2** A homogeneous network is a network  $G = (V, E)$ , where each node  $v_i \in V$  belongs to the same type and each edge  $e_i \in E$  also belong to the same type.

**Definition 3** A attribute network can be defined as  $G_A = (V, E, A, F)$  where  $V$  is the set of vertices,  $E$  is the set of edges,  $A$  is the adjacency matrix and  $F \in R^{n \times k}$ ,  $i$ th row of  $F$  denotes the  $k$  dimensional attribute vector of node  $i$ .

**Definition 4** A heterogeneous network is a network  $G = (V, E)$ , where each node  $v_i \in V$  and each edge  $e_i \in E$ , are associated with mapping functions  $F(v) : V \rightarrow T_v$  and  $f(e) : E \rightarrow T_e$ , where  $T_v$  and  $T_e$  denotes the entity and relationship types respectively.

**Definition 5** A signed network is a network  $G = (V, E)$ ,  $v \in V$ ,  $e \in E$  and for each edge,  $e_{ij} = +1$  or  $e_{ij} = -1$ , denoting a positive link or a negative link between  $v_i$  and  $v_j$ .

**Definition 6** A dynamic network can be defined as a series of snapshots  $G = \{G_1, G_2 \dots G_n\}$  where  $G_i = (V_i, E_i)$  and  $n$  is the number of snapshots.

**Definition 7** First order proximity describes the pair wise proximity between the vertices which is defined using the edge weight  $e_{ij}$  between node  $v_i$  and node  $v_j$ .

**Definition 8** Second order proximity for a pair of nodes  $v_i$  and  $v_j$  is the proximity of neighborhood structures of the nodes  $v_i$  and  $v_j$ .

**Problem 1** Network embedding—Given a network  $G = (V, E)$ , the task is to learn a transformation function  $f : V_i \rightarrow K_i \in R^d$ , where  $d \ll |V|$ , such that  $f$  preserves the first order, second order and higher order proximities of the network.  $d$  defines the number of dimensions of the real valued vector.

## 3 Models of network embedding

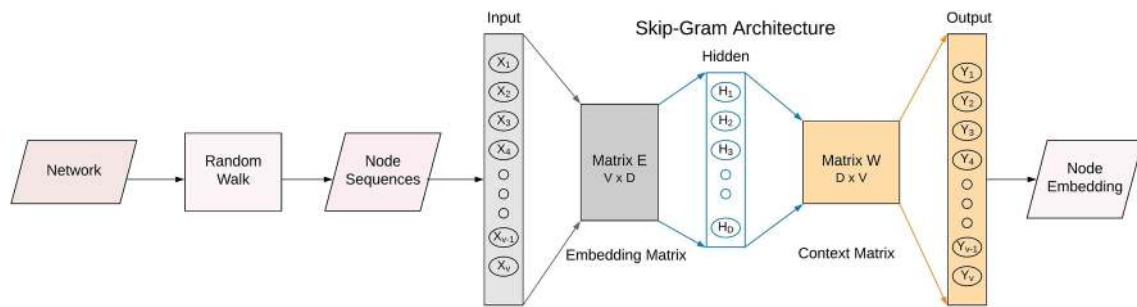
Researches used various models for network embedding which includes both linear and nonlinear dimensionality reduction techniques. Models based on matrix factorization, models that combine random walk sampling with shallow neural networks, and deep neural architectures are the most commonly used models. Other approaches focus on modeling an optimization function based on the structure and the properties to be preserved and solving it using gradient based methods.

### 3.1 Matrix factorization

Network embedding can be considered as a structure-preserving dimensionality reduction process, which assumes that the input data lie in a low dimensional latent space. Network data can be represented in matrix form, which includes adjacency matrix, laplacian matrix, node transition probability matrix and many more. A matrix factorization can be applied on any of these matrices to generate node embeddings. Locally linear embedding [103], Laplacian eigenmaps [6], Structure preserving embedding [107], Graph factorization [2], GraRep [14], HOPE [94] etc. are some among the matrix factorization based methods for network embedding. A detailed survey on these methods can be found in [13, 22, 33]. In this paper, we focus on the methods based on recent advancements in representation learning.

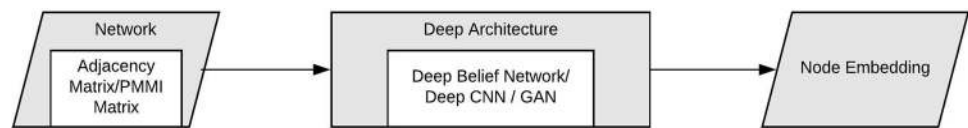
### 3.2 Random walk

Capturing the network structure is a primary concern while generating node embeddings. A random walk is a well-known method which can capture the local structure of the graph. Even if each row in an adjacency matrix corresponds to a node vector which defines the connectivity structure and is analogous to a one hot vector representation, it is very sparse and high dimensional. The word2vec model [85] succeeded in developing a word representation by generating dense low dimensional vectors from



**Fig. 2** A flow diagram of random walk based approaches for network embedding

**Fig. 3** A flow diagram of deep architecture based approaches for network embedding



sparse high dimensional one hot vectors, using a shallow neural network architecture. Word2vec defines two neural architectures, namely continuous bag-of-words model and skip-gram model [86]. The training is done using stochastic gradient descent (SGD) [9]. Word2vec uses two optimization strategies called hierarchical softmax and negative sampling to speed up the training process. Many network embedding methods [29, 42, 92, 96, 101] are inspired from word2vec, which first applies a random walk on the network to generate node sequences that are analogous to sentences in word2vec, followed by using the skip-gram architecture to generate node embeddings. Random walk based approaches try to maximize the probability that the nodes that tend to co-occur on truncated walk lay closer in the embedding space. A general architecture of random walk based procedure for network embedding is shown in Fig. 2.

### 3.3 Deep architecture

The aim of network embedding is to map the nodes from a high dimensional network space to a low dimensional feature space. Some works used specialized neural network models [84, 104], while many others used generalized models over graph-structured data to represent graph in a euclidean space. Network data is inherently non-linear, and using shallow neural network architectures for generating node embedding may result in sub-optimal solutions. Deep neural networks [7, 63, 106] have been successfully used in various domains to learn multiple levels of feature representations from complex and non-linear data. To train large neural networks with more than one hidden layers, many theories and architectures were proposed recently, which includes deep belief networks

(DBN) with greedy layer-wise pre-training [49], deep convolutional neural networks (CNN) [60], long short-term memory networks (LSTM) [41], and generative adversarial networks (GAN) [37]. An autoencoder is a neural architecture which acts as a building block in training deep belief networks. An autoencoder is a three-layer neural network which reconstructs the input vectors at their output layer through a number of non-linear transformations on the input. As an unsupervised feature learning technique, an autoencoder can generate a deep latent representation for the input data. Multiple layers of autoencoders are stacked together to form a stacked autoencoder and it is used as the deep neural architecture for generating node embeddings in many works [15, 39, 129]. Convolutional neural networks, which are very popular in image processing tasks, are not directly applied to graphs, but some works use convolutional architectures [18, 27, 56, 57], which use spectral graph theory [47] to generate node embeddings. Generative adversarial networks (GANs) are deep neural network architectures comprised of two components, a generator and a discriminator, competing one against the other. A few works [24, 130] on network embedding are inspired from GANs. A general architecture of using deep architectures for network embedding is shown in Fig. 3.

## 4 Network representation learning methods

In this section, we review the major works which come under each model with respect to each type of network. The classification of network embedding methods based on different types of networks is depicted in Fig. 4.



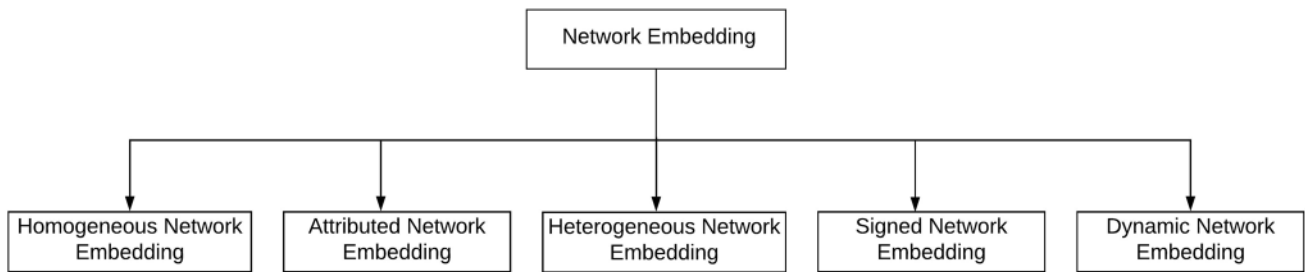


Fig. 4 Network embedding methods based on the types of networks

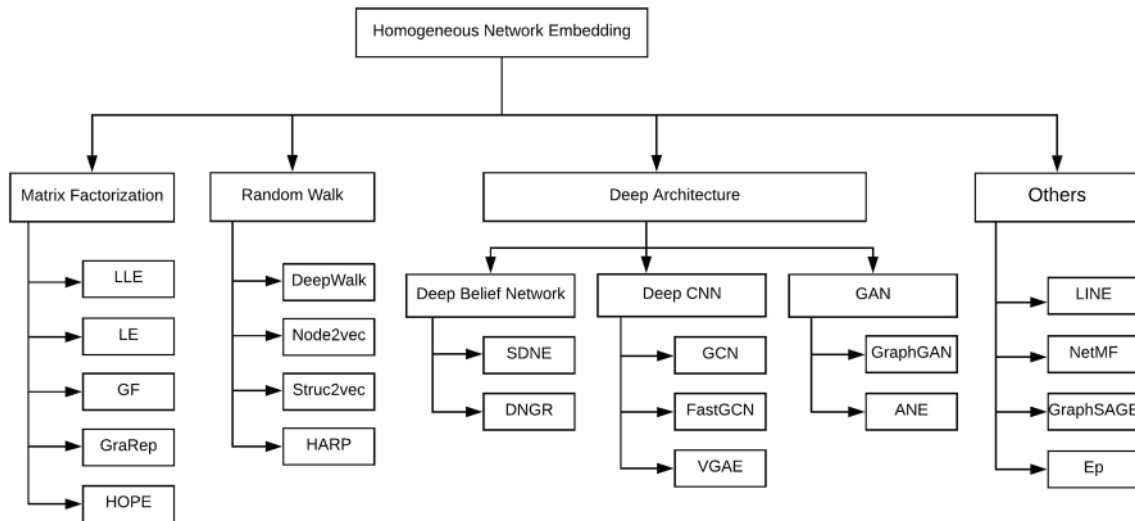


Fig. 5 Homogeneous network embedding methods

#### 4.1 Homogeneous network

Most of the works on network embedding focus on non-attributed, static, unsigned homogeneous networks. Preserving the structural property of the network is the primary objective of homogeneous network embedding methods. Figure 5 shows the major works on homogeneous network embedding which are grouped under major models of network embedding. Table 1 shows a summary of the input, objective function, model used, and properties preserved by some of these methods. Proximity preservation during network embedding is the main aim followed in most of the random walk based methods. Among those, DeepWalk [96] gained a lot of attraction, as it is inspired from the well-studied word2vec algorithm. DeepWalk algorithm involves a two-step process. (1) A truncated random walk on the network to generate a sequence of vertices, which creates an analogy of sentence in word2vec. (2) Using a skip-gram model, which uses a shallow neural network architecture to generate node embeddings. The skip-gram is a generative model whose

objective is to maximize the probability of neighbors in the walk, given the representation of a vertex. For each node  $v_i$ , skip-gram assigns a current  $d$  dimensional representation,  $\phi(v_i) \in R^d$  and maximizes the co-occurrence probability of its neighbors in the walk to update this representation. The optimization becomes,

$$\max \log P(v_{i-w}, \dots, v_{i+w} | \phi(v_i)) \quad (1)$$

where  $v_{i-w}, \dots, v_{i+w}$  denotes the neighbors of  $v_i$  in the node sequence, and  $w$  is the context size. Computing the softmax at the output layer of skip-gram is computationally expensive and DeepWalk approximates softmax using two strategies, hierarchical softmax and negative sampling. These strategies reduces the time complexity of skip-gram model and speed up the training process. As random walk being a sampling strategy, the time complexity to perform random walk is linear w.r.t the number of edges. The complexity of the skip-gram architecture is proportional to  $O(C(D + D \log_2(V)))$ , where  $C$  is the context window size,  $D$  is the number of dimensions and  $\log_2(V)$  is the time to build the hierarchical softmax over  $V$  vertices. DeepWalk is

**Table 1** Summary of homogeneous network embedding methods

Method	Input	Objective function	Model	Properties preserved
DeepWalk [96]	Adjacency list/edge list	$\max \log P(v_{-w}, \dots, v_{+w}   \phi(v_i))$	Skip-gram with hierarchical softmax	Local graph structure
Node2vec [42]	Edge list	$\max_{v_i \in W, -w \leq t \leq w} \sum \log P(v_t   \phi(v_i))$	Skip-gram with negative sampling	Local graph structure and structural equivalence
HARP [17]	Edge list	Same as that of DeepWalk/Node2vec/LINE	Skip-gram with hierarchical softmax	Higher order proximities
SDNE [129]	Adjacency matrix/edge list	$L = \gamma' \ (Y - X) \odot B\ ^2 + \alpha' \sum_{i=1}^n a_i \ \phi_i - \phi_j\ _2^2 + \frac{\lambda}{2} \sum_{k=1}^K (\ W^{(k)}\ _F^2 + \ \hat{W}^{(k)}\ _F^2)$	Stacked autoencoders	First and second order proximities
DNGR [15]	PPMI matrix	$\arg \min_{\theta_1, \theta_2} \sum_{i=1}^n \ x_i - g_{\theta_1}(f_{\theta_2}(y_i))\ ^2$	Stacked denoising autoencoders	Non-linearity in network data
GCN [56]	Adjacency matrix	$H_{l(l+1)} = \sigma(\hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}} H_{(l)}) W_{(l)}$	Graph convolutional network	Local graph structure
GraphGAN [130]	Edge list	$\min_{\theta_G} \max_{\theta_D} V(G, D) = \sum_{c=1}^V (E_{v \sim p_{\text{true}}(v_c)} [\log D(v, v_c; \theta_D)] + E_{v \sim G(v; \theta_G)} [\log(1 - D(v, v_c; \theta_D))])$	Graph softmax as generator; any discriminative model as discriminator	Local graph structure

parallelizable and can be implemented without the knowledge of entire graph, which makes it suitable for large-scale machine learning. DeepWalk motivated many subsequent works [17, 24, 29, 42, 92, 95, 101] and also acted as a baseline for various works in the area of network representation learning. Walklet [97] modified the random walk used in DeepWalk by explicitly preserving the proximities between vertices during the random walk, and showed that the multi-scale representations thus generated can improve the performance of multi-label classification task. Max-Margin DeepWalk [121] and Discriminative Deep Random Walk [69] extended DeepWalk by associating classification objective with embedding objective and thereby demonstrated the performance improvement of the multi-label classification task.

The quality of the network embedding can be further improved by preserving the structural equivalence of the nodes along with the proximity information. Node2vec [42] works towards this goal by performing a biased random walk which can provide more flexibility in exploring neighborhood compared to DeepWalk. During random walk, node2vec uses two sampling strategies, breadth-first search (BFS) and depth-first search (DFS) which traverse the search space by exploring both community structures and structurally equivalent nodes in the network. Based on the random walk sequence, node2vec extends skip-gram architecture to optimize the objective function,

$$\max \sum_{v_t \in W_v} \sum_{-w \leq t' \leq w} \log P(v_{t'} | \phi(v_t)) \quad (2)$$

where  $v_t$  is the node taken from the random walk sequence  $W_v$ ,  $v_{t'}$  is the neighbor node of node  $v_t$  within the window  $w$ , and  $\phi(v_t) \in \mathbb{R}^d$  is the feature representation of the node  $v_t$ . Node2vec incurs additional space and time complexity over deepwalk as it involves BFS and DFS search during random walk. Node2vec can preserve the structural equivalence of nodes in the network but is limited by the size of the context window. Struc2vec [101] aims at preserving structural equivalence to a better extent by computing the structural similarity between each pair of vertices in the network. Struc2vec constructs a multilayer network, where each layer denotes a hierarchy in measuring the structural similarity, and then applies random walk sampling followed by skip-gram learning on the multilayer graph to generate the embedding of each vertex.

Preserving higher order structural patterns in large-scale networks is a challenging issue. HARP [17] is a meta-strategy that can achieve this goal. HARP can be used to improve the state-of-the-art NRL algorithms [42, 96, 115] so as to avoid these algorithms to get stuck in local optima, as these models rely on non-convex optimization, solved

by SGD. HARP progresses through three steps—graph coarsening, embedding and representation refinement. In coarsening, a large network is divided into smaller networks by preserving the original structure using two strategies, edge collapsing and star collapsing. The embedding algorithm is applied to the coarsest graph and the embedding is generated. The last step is to prolong and refine the network from coarsest to finest. To perform refinement, HARP uses two strategies, multilevel hierarchical softmax and multilevel negative sampling. The overall time complexity of HARP (with deepWalk) is  $O(\gamma|V|)$  where  $\gamma$  is the number of walks and  $V$  is the number of vertices. Experiments show that the HARP extension can improve the performance of DeepWalk, LINE, and Node2vec over multi-label classification.

Network structure is inherently non-linear, and using a shallow neural network for network embedding may lead to suboptimal solutions. SDNE [129] addresses this challenge by using a deep architecture, build with stacked autoencoders, to generate network embeddings. SDNE deploys a deep belief network, implements multiple layers of non-linear functions, and map the data into a non-linear feature space. In order to maintain the structure-preserving property and to address sparsity, SDNE trains a joint optimization function (shown as equation 3) which preserves the first order and second order proximities. This function preserves the second order proximity using stacked autoencoders and the first order proximities using Laplacian Eigenmaps.

$$L = \gamma' ||(Y - X) \odot B||^2 + \alpha' \sum_{i=1}^n \sum_{j=1}^n a_{ij} ||\phi_i - \phi_j||_2^2 + \frac{\lambda}{2} \sum_{k=1}^K \left( ||W^{(k)}||_F^2 + ||\hat{W}^{(k)}||_F^2 \right) \quad (3)$$

Here  $X$  and  $Y$  are the input and reconstructed data. The model minimizes the reconstruction error to capture the global information.  $B$  has been used to handle the sparsity of the adjacency matrix  $a_{ij}$ .  $\phi_i$  and  $\phi_j$  represent the feature representation of node  $i$  and  $j$  and  $W^{(k)}$  is the hidden layer weight matrix for the autoencoder. The time complexity of SDNE is  $O(ncdi)$ , where  $n$  is the number of vertices,  $d$  is the embedding dimension,  $c$  is the average degree of the network and  $i$  is the number of iterations.

In language modeling, alternate method to generate word vectors is to find the low dimensional linear projections from positive point-wise mutual information matrix (PPMI) [12, 68] of words and their contexts. DNGR [15] is inspired from [68], which first uses a random surfing model to generate a probabilistic co-occurrence matrix that captures the graph structure information. The PMMI matrix is then

calculated from the co-occurrence matrix. Instead of applying singular value decomposition (SVD) as in [14], DNGR applies a stacked denoising autoencoder on PMMI matrix which learns a non-linear function to map high dimensional vertex vectors into low dimensional node embeddings. The authors of DNGR claim that, using the probabilistic co-occurrence matrix is well suited for weighted networks and is less computationally expensive compared to sampling-based methods [42, 96]. The time complexity of DNGR is defined to be linear w.r.t. the number of vertices in the graph. The objective function of DNGR is defined as

$$\arg \min_{\theta_1, \theta_2} \sum_{i=1}^n \|x_i - g_{\theta_2}(f_{\theta_1}(y_i))\|^2 \quad (4)$$

where  $x_i$  is the  $i$ th instance,  $y_i$  is the corrupted input data of  $x_i$ , and  $f_{\theta_1}$  and  $g_{\theta_2}$  are the encoding and decoding functions of the autoencoder respectively.

A few efforts are made to apply some variants of CNN [27, 47], to perform representation learning on networks. GCN [56] is one such approach whose goal is to learn a function from the network which takes as input (1)  $N \times K$  feature matrix, where  $N$  is the number of nodes and  $K$  is the number of input features. (2) An adjacency matrix  $A$ . The GCN produces an output  $Z$  which is an  $N \times D$  matrix, where  $D$  is the number of dimensions per node. GCN uses the layer-wise propagation rule

$$H_{(l+1)} = \sigma\left(\hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}} H_{(l)} W_{(l)}\right) \quad (5)$$

where  $W_l$  denote the weight matrix of  $l$ th network,  $\hat{A} = A + I$ ,  $D$  is the diagonal node degree matrix and  $H_{(0)} = X$ , the matrix of node attributes. The authors interpreted GCN as a generalized version of the weisfeiler-Lehman algorithm on graphs. The complexity of the convolution operation is  $O(efc)$  where  $e$  is the number of edges,  $f$  is the number of filters and  $g$  is the node dimension. Fast-GCN [18] is an enhancement over GCN where the authors interpret graph convolutions as integral transforms of embedding functions under probability measures, and uses Monte Carlo approaches to consistently estimate the integrals. Parametric graph convolution [119] is another enhancement over GCN which generalizes a convolutional filter by adding a hyper-parameter that influences the filter size, and thereby improves the performance of GCN. Variational graph autoencoder (VGAE) [57] is another work which uses variational autoencoder to learn latent representations from undirected graphs. The authors demonstrated this model using a graph convolutional network (GCN) encoder and a simple inner product decoder.

GraphGAN [130] directly follows the GAN architecture which tries to learn two models: (1) a generator which approximates the underlying connectivity distribution

and generates fake vertex pairs to fool the discriminator. (2) a discriminator that tries to distinguish the vertex pairs that is generated by the generator from the real ones. The objective of the discriminator is to maximize the logarithmic probability of assigning correct labels to real and generated samples. The generator objective is to minimize the logarithmic probability that the discriminator correctly identifies the samples generated by the generator. A sigmoid and softmax function are used as the discriminator and generator function respectively. The work also proposes an alternate method called graphsoftmax, which can improve the performance of softmax. The objective function of GraphGAN is modeled as a two-player minmax game with cost function

$$\min_{\theta_G} \max_{\theta_D} V(G, D) = \sum_{c=1}^V (E_{v \sim p_{true}(\cdot|v_c)} [\log D(v, v_c; \theta_D)] + E_{v \sim G(\cdot|v_c; \theta_G)} [\log(1 - D(v, v_c; \theta_D))]) \quad (6)$$

Here the generator  $G$  tries to generate vertices which resembles vertex  $v_c$ 's neighbors by approximate the underlying true connectivity distribution  $p_{true}(v|v_c)$ , and the discriminator  $D$  tries to discriminate the true neighbor of  $v_c$  from those generated by  $G$  by finding the probability of an edge to exist between  $v$  and  $v_c$  which is represented as  $D(v, v_c; \theta_D)$ . By minimizing and maximizing the cost function, the optimal parameters for  $D$  and  $G$  can be learned, and GraphGAN uses policy gradient ascent procedure to learn the parameters. The time complexity of each iteration of GraphGAN is  $O(V \log V)$ , where  $V$  is the number of vertices.

ANE [24] proposes a different approach which uses adversarial learning as a regularizer to learn more robust network representations. ANE employs a structure-preserving component and an adversarial learning component. For structure preservation, ANE uses a method called inductive DeepWalk (IDW). IDW perform random walk using PMMI matrix to explore the neighborhood, and optimizes a parameterized generator function to generate embeddings. The adversarial learning component consists of a generator and a discriminator. It shares the generator function with the structure-preserving component. Initially, the discriminator is trained to separate the prior samples from the embedding vectors. The parameters of the generator are then updated so as to fool the discriminator and thereby performing regularization on the embedding generated by the structure-preserving component.

#### 4.1.1 Other works

**LINE**—The objective of LINE [115] is to preserve first order and second order proximity during embedding. LINE first calculates the joint probability between two vertices in two ways, one using edge weights and other using node vectors.



To preserve first order proximity, LINE defines an objective function to minimize the distance between two probability distributions. Objective function to preserve second order proximity is also defined in a similar way. LINE uses edge sampling strategy to speed up the computations.

**NETMF**—The works [42, 96, 115] lacked through theoretical analysis and the same is provided by [100]. The work reveals that, all these methods are essentially performing implicit matrix factorization. Analysing closed form matrices of all the methods, [100] eventually discusses the relationship between these methods and their connection with the Graph Laplacian. The authors also propose a method called NETMF which explicitly factorize the closed form implicit matrix of DeepWalk using singular value decomposition (SVD) and generates the node embeddings.

**GraphSAGE**—It [45] is an inductive representation learning method which is suitable for large graphs. Instead of training individual embeddings for each node, GraphSAGE learns a function that generates node embeddings by sampling and aggregating features from the nodes local neighborhood.

**Ep**—Embedding propagation (Ep) [35] is network representation method inspired from label propagation. EP sends forward and backward messages between neighboring nodes. Forward messages contain label representations and backward messages contain gradients that result from aggregating the label representations and applying a reconstruction loss. Node representations are computed from label representations.

## 4.2 Attributed network embedding

In most of the real-world networks, nodes or edges are associated with single or multiple attributes which provide some semantic information. In this section, we will cover some methods which perform network embedding on such attributed networks [98, 99]. Figure 6 shows the major works under attributed network embedding. Table 2 shows a summary of the input, objective function, model used, and properties preserved by some of these methods.

Nodes of the network may have text associated with it. TADW [139] aims to embed networks by using the structural information and the text information associated with the nodes. The work proves the equivalence of DeepWalk and closed form matrix factorization, and creates a PMMI matrix using vertex–context pairs for further processing. TADW performs inductive matrix completion [90] to associate text features into PMMI matrix, and low-rank matrix factorization on the resultant matrix to generate the network embedding. The objective function of TADW is stated as

$$\min_{W,H} \|M - W^T H T\|_F^2 + \frac{\lambda}{2} (\|W\|_F^2 + \|H\|_F^2) \quad (7)$$

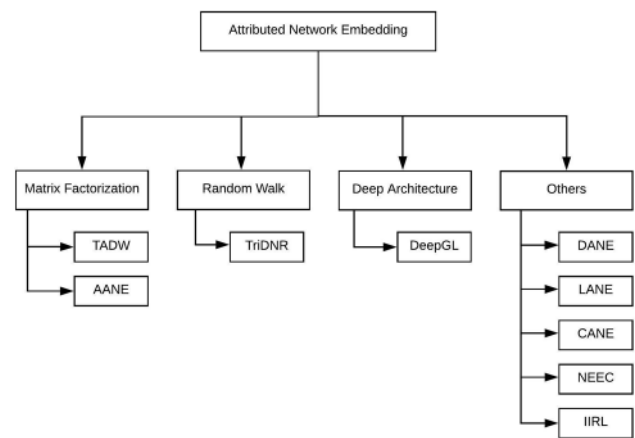


Fig. 6 Attributed network embedding methods

where  $M$  and  $T$  are the word-context matrix and text feature matrix respectively, and  $\min_{W,H} \|M - W^T H T\|_F^2$  represents the low rank matrix decomposition of matrix  $M$ . The complexity of each iteration of minimizing  $W$  and  $H$  is  $O(n_0(M)k + |V|f_t k + |V|k^2)$  where  $n_0(M)$  indicates the number of non-zero entries of  $M$ , and  $k$  denotes the low rank of  $M$ .

Accelerated attributed network embedding (AANE) [51] is another approach which uses connectivity information and attribute information to perform network embedding. AANE nodes a joint optimization function with two components, a strategy based on spectral clustering to preserve node proximities and a matrix factorization framework to approximate the attribute affinity matrix. Further, the authors have provided a distributed algorithm to solve the optimization objective in an efficient manner. The loss function of AANE is modeled as

$$\min_H \|S - H H^T\|_F^2 + \lambda \sum_{(i,j) \in \epsilon} w_{ij} (\|h_i - h_j\|_F^2) \quad (8)$$

Here  $S$  represent the attribute affinity matrix,  $H$  the embedding matrix and  $h_i$  and  $h_j$  are the vector representations of node  $i$  and node  $j$ .  $\min_H \|S - H H^T\|_F^2$  component preserve the node attribute proximity and  $\lambda \sum_{(i,j) \in \epsilon} w_{ij} (\|h_i - h_j\|_F^2)$  component preserve the network structure proximity. The time complexity of AANE is  $O(nN_A + n^2)$ , where  $N_A$  is the number of nonzero in attribute affinity matrix  $A$  and  $n$  is the number of nodes in the network.

In real-world networks like citation networks, papers may have text associated with it, the category information that the paper belongs to, and a reference link to other papers. Such networks can be modeled as graphs with node structure, content, and labels. TriDNR [95] aims at generating embedding by exploiting all these three levels. It uses the idea from DeepWalk and paragraph vector algorithm [62] to embed node, text and label information.

Table 2 Summary of attributed network embedding methods

Method	Input	Objective function	Model	Properties preserved
TADW [139]	Edge list and feature matrix	$\min_{W,H} \ M - W^T H\ _F^2 + \frac{\lambda}{2} (\ W\ _F^2 + \ H\ _F^2)$	Low-rank matrix factorization	Network structure and text feature similarity
AANE [51]	Adjacency and attribute matrix	$\min_H \ S - H H^T\ _F^2 + \lambda \sum_{(i,j) \in E} w_{ij} (\ h_i - h_j\ _F^2)$	Matrix factorization	Node proximities and attribute similarity
TriDNR [95]	Adjacency and attribute matrix	$\max(1 - \alpha) \sum_{i=1}^N \sum_{s \in S} \sum_{-b \leq j \leq b, j \neq 0} \log P(v_{i+j} v_i) + \alpha \sum_{i=1}^N \sum_{-b \leq j \leq b} \log P(w_j v_i) + \alpha \sum_{i=1}^{ L } \sum_{-b \leq j \leq b} \log P(w_j c_i)$	Extension of skip-gram with hierarchical softmax	Node proximities and attribute similarity
DeepGL [102]	Adjacency and attribute matrix	$\arg \max_{x_i \in X} (K(y, x_i) - \beta \sum_{x_j \in X} K(x_i, x_j))$	Deep neural architecture	Node proximities and attribute similarity

TriDNR models a joint optimization function which learns inter-node, node-content and label-content correlations, and the training is done using SGD. It also uses hierarchical softmax to speed up the computations. The objective function of TriDNR is to maximize the log-likelihood

$$\begin{aligned} & \max(1 - \alpha) \sum_{i=1}^N \sum_{s \in S} \sum_{-b \leq j \leq b, j \neq 0} \log P(v_{i+j}|v_i) \\ & + \alpha \sum_{i=1}^N \sum_{-b \leq j \leq b} \log P(w_j|v_i) \\ & + \alpha \sum_{i=1}^{|L|} \sum_{-b \leq j \leq b} \log P(w_j|c_i) \end{aligned} \tag{9}$$

The first component of the equation to is maximize the likelihood of the neighboring nodes given current node  $v_i$ , the second component maximizes probability of observing contextual words given the current node  $v_i$ , and the third component maximizes the likelihood of observing the words given a class label  $c_i$ .  $\alpha$  is balance parameter to control the proportion of network structure, text, and label information.

DeepGL [102] is a deep architecture which performs hierarchical representation learning on attributed networks. DeepGL first generates a set of base features by performing graphlet decomposition on higher order network motifs(graphlets). DeepGL learns a set of relational feature operations, which when applied on the base features generates a set of higher level features. At each layer of the deep architecture of DeepGL, features from lower order subgraphs are combined using composition of relational feature operations to generate higher order subgraph patterns. DeepGL is designed to be effective for network-based transfer learning tasks. The optimization function of DeepGL is stated as

$$\arg \max_{x_i \in X} \left\{ K(y, x_i) - \beta \sum_{x_j \in X} K(x_i, x_j) \right\} \tag{10}$$

which aims to find a set of features  $x_i$  that maximizes its similarity to the label  $y$  and minimizes the similarity between the features  $x_i$  and  $x_j$  in the collection. The complexity of generating node embeddings with the DeepGL is  $O(F(M + NF))$ , where  $N$ ,  $M$ , and  $F$  are the number of nodes, edges, and node features respectively.

The GCN methods presented in the previous section can also deal with attributed networks.

#### 4.2.1 Other works

DANE—It [70] aims at generating representations from a network with structure and attribute information. DANE provides an online learning model, which extends the basic design to a distributed environment. DANE takes as input an adjacency matrix and a user feature matrix, and

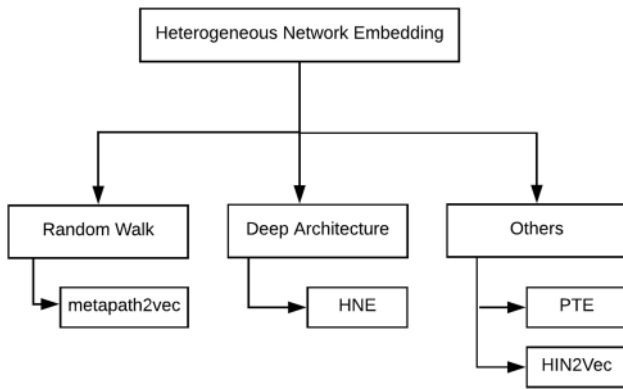


Fig. 7 Heterogeneous network embedding methods

generates two embedding  $E_a$  and  $E_x$ , using spectral clustering based methods. Further, DANE generates a single embedding representation by maximizing the correlation between  $E_a$  and  $E_x$ . DANE uses matrix perturbation theory to update  $E_a$  and  $E_x$ , and to generate the updated embedding  $E$ .

**LANE**—Label informed attributed network embedding (LANE) [52] affiliates labels with the attributed network and maps the network into a low-dimensional representation by modeling their structural proximities and correlations.

**CANE**—Context-aware network embedding (CANE) [122] is another attributed network embedding which learns various context-aware embeddings for a vertex according to the neighbors it interacts with.

**NEEC**—NEEC [53] aims at improving the attributed network embedding by learning and incorporating the expert cognition into the embedding objective.

**IIRL**—IIRL [138] uses two terminologies, structure close links and content close links to define the topological and attribute similarity between nodes. A joint optimization function is defined to preserve the proximity of structure-close and content-close links in the embedding space, and the training is done using a gradient based algorithm.

### 4.3 Heterogeneous network embedding

Typically, some network mining tasks demand the data to be modeled as heterogeneous networks [111] that involve nodes and edges of different types. For example, a citation network can be modeled as a heterogeneous network with authors, papers, and venue as nodes and relationship between these types as edges. In this section, we will cover some methods which perform network embedding on heterogeneous networks. Figure 7 shows the major works on heterogeneous network embedding and Table 3 shows a summary of these methods.

Table 3 Summary of heterogeneous network embedding methods

Method	Objective function	Model	Properties preserved
Metapath2vec [29]	$\arg \max_{\theta} \sum_{v \in V} \sum_{r \in \mathcal{R}_v} \sum_{c_i \in \mathcal{M}(v)} \log P(c_i   v; \theta)$	Heterogeneous skip-gram with heterogeneous negative sampling	Structural and semantic correlations between heterogeneous nodes
HNE [16]	$\min_{U, V} \frac{1}{N_H} \sum_{v_i, v_j \in V} L(x_i, x_j) + \lambda_3 (\ U\ _F^2 + \ V\ _F^2) + \frac{\lambda_2}{N_H} \sum_{v_i, v_j \in V} L(z_i, z_j) + \frac{\lambda_1}{N_H} \sum_{v_i, v_j \in V} L(x_i, z_j)$	Deep neural architecture	Non-linear relationships between heterogeneous nodes
PTE [114]	$\text{minimize} - \sum_{(i,j) \in E_{sw}} w_{ij} \log p(v_i   v_j) - \sum_{(i,j) \in E_{wd}} w_{ij} \log p(v_i   d_j) - \sum_{(i,j) \in E_{ww}} w_{ij} \log p(v_i   v_j)$	Stochastic gradient descent with edge and negative sampling	Semantic relationships between heterogeneous nodes
HIN2Vec [32]	$\max_O \sum_{x, y, r \in D} \log O_{x, y, r}(x, y, r)$	Neural network architecture	Heterogeneous relationships between nodes

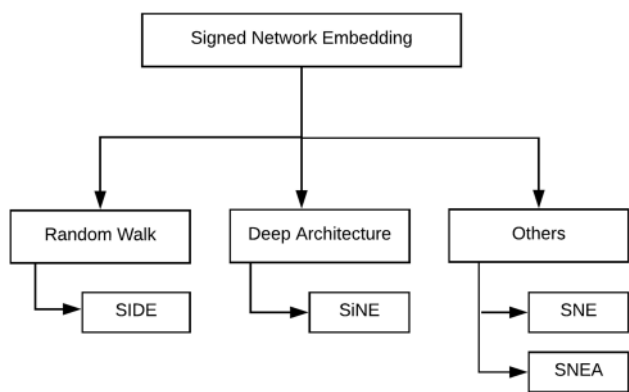


Fig. 8 Signed network embedding methods

Metapath2vec [29] is an extension of random walk and skip-gram based methods which can be applied to heterogeneous networks. A meta path [112] is a path that can be represented in the form  $V_1 \xrightarrow{R_1} V_2 \xrightarrow{R_2} \dots V_t \xrightarrow{R_t} V_{t+1} \dots \xrightarrow{R_{k-1}} V_k$  where  $R = R_1 \circ R_2 \circ R_3 \circ \dots \circ R_{k-1}$  defines the composition relations between node types  $V_1$  and  $V_k$ . Metapath2vec performs a metapath based random walk through the heterogeneous network, and generate paths which can capture both the structural and semantic relationship between different types of nodes. The resulting paths are fed to a heterogeneous skip-gram model which can learn the representation of nodes by maximizing the probability of heterogeneous context nodes, given the input node. The objective function of heterogeneous skip-gram is stated as

$$arg \max_{\theta} \sum_{v \in V} \sum_{t \in T_v} \sum_{c_t \in N_{t(v)}} \log P(c_t | v; \theta) \tag{11}$$

where  $t$  denotes the node type,  $N_{t(v)}$  denotes neighborhood of node  $v$ , and  $P(c_t | v; \theta)$  is a softmax function which calculates the probability of co-occurrence of context-input pairs. The time complexity of metapath2vec is same as that of deepwalk as both uses the skip-gram architecture for learning node representations. The work also discusses an algorithm called metapath2vec++, which provides heterogeneity in negative sampling by maintaining separate multinomial distributions for each node type in the output layer of the skip-gram model, and thereby provides more efficiency and accuracy in representation.

The main aim of HNE [16] is to map the multimodal objects in the heterogeneous network to a common space such that the similarity between the objects can be computed. HNE considers a heterogeneous network with text–text, text–image, and image–i–image interactions as input. Text and image data are transformed into  $d$ -dimensional vectors and are mapped to a latent space using linear transformations. An objective function is modeled to minimize the distance between the objects if they

Table 4 Summary of signed network embedding methods

Method	Objective function	Model	Properties preserved
SIDE [55]	$maximize \sum_{(u,v) \in E} [-\log(P(u,v))] + \sum_{j=1}^n -\log P(u,v_j)$	Skip-gram with negative sampling	Network structure with signed edges and directions
SiNE [133]	$\min_{X_{x_0}, \theta} \frac{1}{C} \sum_{(x_i, x_k) \in P} \max(0, f(x_i, x_k) + \delta - f(x_i, x_j)) + \sum_{(x_i, x_0) \in P_0} \max(0, f(x_i, x_0) + \delta_0 - f(x_i, x_j)) + \alpha(\ X\ _F^2 + \ x_0\ _2^2)$	Deep neural architecture	Local proximities, non-linearity, and signed edges
SNE [140]	$\max_{v \in V} \log \frac{\exp(s(v, h))}{\sum_{v' \in V} \exp(s(v', h))}$	Log-bilinear with optimization using adagrad and sampled softmax	Network structure with signed edges

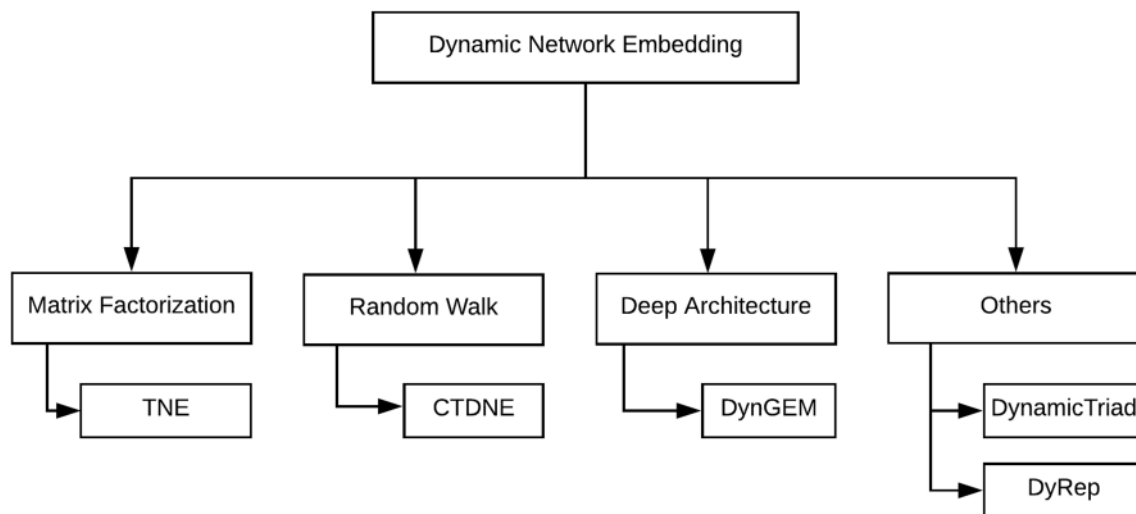


Fig. 9 Dynamic network embedding methods

are topologically connected. The loss function of HNE is stated as

$$\begin{aligned}
 \min_{U,V} & \frac{1}{N_{II}} \sum_{v_i, v_j \in V_I} L(x_i, x_j) + \lambda_3 (||U||_F^2 + ||V||_F^2) \\
 & + \frac{\lambda_1}{N_{TT}} \sum_{v_i, v_j \in V_T} L(z_i, z_j) \\
 & + \frac{\lambda_2}{N_{IT}} \sum_{v_i \in V_I, v_j \in V_T} L(x_i, z_j)
 \end{aligned} \tag{12}$$

where the first component represents the loss w.r.t. text to text similarity, second component represents the loss w.r.t. image to image similarity and the third component represents the loss w.r.t. text to image similarity.  $N_{II}$ ,  $N_{TT}$  and  $N_{IT}$  are the numbers of the three types of links, and  $\lambda_1$ ,  $\lambda_2$  and  $\lambda_3$  are the three balancing parameters. HNE further proposes a deep architecture which can map different modalities into a common subspace, and can construct the feature representation. A CNN is used to learn the image features and a fully connected layer is used to learn the text features. A linear embedding layer is used to map the input to a common subspace. A prediction layer is used to calculate the loss function and the training is done using SGD.

The authors of LINE [115] extended their network embedding approach on the homogeneous network to a heterogeneous network using PTE [114]. PTE constructs a heterogeneous text network by combining a word-word network, a word-document network, and a word-label network. PTE then apply LINE to embed the three bipartite networks. PTE further models a joint optimization function which can collectively embed the three bipartite networks to generate a single heterogeneous text network embedding. The loss function of PTE is stated as

$$\begin{aligned}
 \text{minimize} & - \sum_{(i,j) \in E_{ww}} w_{ij} \log p(v_i | v_j) - \sum_{(i,j) \in E_{wd}} w_{ij} \log p(v_i | d_j) \\
 & - \sum_{(i,j) \in E_{wl}} w_{ij} \log p(v_i | l_j)
 \end{aligned} \tag{13}$$

The first, second and third term of the equation is to minimize the negative log-likelihood of co-occurrence of word-word pair, word-document pair and word-label pair respectively. The authors provided two approaches (1) a joint training and (2) a pre-training and fine tuning to perform the learning process.

HIN2Vec [32] is another work which uses meta path based approach for representation learning in heterogeneous information networks. Initially, hin2vec proposes a conceptual neural network architecture which is trained to learn the relationship between the nodes, by placing the possible meta paths at the output layer. The objective function of HIN2Vec is

$$\max_O \sum_{x,y,r \in D} \log O_{x,y,r}(x, y, r) \tag{14}$$

The function takes as input, a pair of nodes  $x$  and  $y$ , and a relationship  $r$ , and tries to maximize the prediction probability whether the relationship exists between  $x$  and  $y$ . Further, the authors provide an enhanced neural network architecture which can learn the node embedding and meta path embedding during the training process.

#### 4.4 Signed networks

Signed networks [67, 116] are part of real social systems where the relationship between entities can be either positive or negative. In this section, we will cover some methods which perform network embedding on signed



Table 5 Summary of dynamic network embedding methods

Method	Objective function	Model	Properties preserved
TNE [147]	$\arg \min_{z_1, z_2, \dots, z_t} \sum_{t=1}^t \ G_T - Z_T Z_T^T\ _F^2 + \lambda \sum_{t=1}^t \sum_u (1 - Z_T(u) Z_{T-1}(u)^T) \text{ subject to : } \forall u, \Gamma, Z_T \geq 0, Z_T(u) Z_{T-1}(u)^T = 1$	Non-negative matrix factorization	Network structure, temporal smoothness and latent homophily
CTDNE [92]	$\max_f \log P_r(W_T = \{v_{-w}, \dots, v_{+w}\} \setminus v_i   f(v_i))$	Skip-gram architecture	Network structure and temporal dependencies
DynGEM [39]	$L = \gamma \  (Y - X) \odot B \ ^2 + \alpha' \sum_{i=j=1}^n a_{ij} \ \phi_i - \phi_j\ _2^2 + \frac{\lambda}{2} \sum_{k=1}^K (\ W^{(k)}\ _1 + \ \hat{W}^{(k)}\ _1) + \frac{\lambda}{2} \sum_{k=1}^K (\ W^{(k)}\ _F^2 + \ \hat{W}^{(k)}\ _F^2)$	Stacked autoencoders	Network structure, non-linearity, and stability
Dynamic-Triad [146]	$\sum_{t=1}^T (\alpha_{s_{h,1}}^t + \beta_0 l_{tr,2}^t) + \beta_1 \sum_{t=1}^{T-1} \sum_{i=1}^N \ u_i^{t+1} - u_i^t\ _2^2$	Function minimization using stochastic gradient descent with adagrad	Network structure, temporal smoothness, and social homophily

networks. Various works on signed network embedding are listed in Fig. 8 and a summary of these methods is shown in Table 4.

SIDE [55] is a network embedding method for signed directed networks. SIDE follows the random sampling strategy and hierarchical optimization, which is well exploited by language models. SIDE performs a truncated random walk on a signed directed network, and generates positive and negative node pairs based on structural balance theory. Then, SIDE models an optimization function which can be stated as

$$\begin{aligned} \text{maximize } & \sum_{(u,v) \in D} [-\log(P(u, v)) + \sum_{j=1}^n -\log P(u, v_j)] \\ & + \frac{\lambda}{2} (\|b^{in,+}\|^2 + \|b^{in,-}\|^2 + \|b^{out,+}\|^2 + \|b^{out,-}\|^2) \end{aligned} \tag{15}$$

The function tries to find the parameters that maximize the likelihood  $p(u, v)$  between two nodes  $u$  and  $v$  such that the likelihood value is high for positively connected nodes and low for negatively connected nodes. The latter part of the objective function regularizes the bias terms in the function. The time complexity SIDE is linear w.r.t. the number of nodes in the network.

SiNE [133] is a deep learning architecture for signed network embedding. It uses the structural balance theory which assumes that, a node is more similar to a node with a positive link than to a node with a negative link. SiNE first defines a similarity function between the d-dimensional representations of nodes, and models an optimization function to learn the parameters of the similarity function which is stated as

$$\begin{aligned} \min_{X, X_0, \theta} & \frac{1}{C} \left[ \sum_{(x_i, x_j, x_k) \in P} \max(0, f(x_i, x_k) + \delta - f(x_i, x_j)) \right. \\ & \left. + \sum_{(x_i, x_j, x_0) \in P_0} \max(0, f(x_i, x_0) + \delta_0 - f(x_i, x_j)) \right] \\ & + \alpha(R(\theta) + \|X\|_F^2 + \|x_0\|_2^2) \end{aligned} \tag{16}$$

where  $P$  defines the set of triplets  $(v_i, v_j, v_k)$  where  $v_i$  and  $v_j$  have a positive link while  $v_i$  and  $v_k$  have a negative link, and  $P_0$  defines set of triplets  $(v_i, v_j, v_0)$  where  $v_i$  and  $v_j$  have a positive link while  $v_i$  and  $v_0$  have a negative link.  $C$  is the size of the training data and  $\theta$  is the set of parameters to learn. SIDE uses a deep neural architecture with two hidden layers to optimize the objective function.

SNE [140] is a log-bilinear model [88] for generating embedding from the signed network. Given a path, SNE tries to predict the embedding of node  $v$  by linearly combining the feature vectors nodes in the path with corresponding signed weight vectors. A scoring function is used

to measure the similarity between actual and predicted representation. The optimization objective is

$$\max \sum_{v \in V} \log \frac{\exp(s(v, h))}{\sum_{v^i \in V} \exp(s(v^i, h))} \tag{17}$$

The objective function is to maximize the logarithmic likelihood of a target node  $v$ , generated by a path of nodes  $h$  and their edge types, using a softmax function. Attribute signed network embedding [132] is another work which addresses network embedding on the signed network with an attribute associated with nodes. Initially, SNEA defines two optimization functions to generate embedding, one for modeling user attributes and other for modeling signed networks using structural balance theory. Then it models a joint optimization function by combining these components, and the training is done using gradient descent.

### 4.5 Dynamic network embedding

Many real-world networks are dynamic and will evolve over time [66, 109]. Between adjacent snapshots, new nodes and edges may be added and existing ones may be lost. In this section, we will cover some methods which perform network embedding on dynamic networks. Figure 9 shows the major works on dynamic network embedding and Table 5 describes a summary of these methods.

The research done by [147] aims at developing a temporal latent space model that can predict links over time based on a sequence of previous graph snapshots. The authors first model a quadratic loss function to learn the temporal latent space from a dynamic social network which is stated as

$$\begin{aligned} & \arg \min_{z_1, z_2, \dots, z_t} \sum_{\Gamma=1}^t \|G_\Gamma - Z_\Gamma Z_\Gamma^T\|_F^2 \\ & + \lambda \sum_{\Gamma=1}^t \sum_u (1 - Z_\Gamma(u) Z_{\Gamma-1}(u)^T) \\ & \text{subject to : } \forall u, \Gamma, Z_\Gamma \geq 0, Z_\Gamma(u) Z_\Gamma(u)^T = 1 \end{aligned} \tag{18}$$

where the first term denotes the matrix factorization of adjacency matrix representations of the network snapshots  $G_\Gamma$  and the second term  $1 - Z_\Gamma(u) Z_{\Gamma-1}(u)^T$  penalizes node  $u$  for a sudden change in its latent position. The objective of the loss function is to maintain the temporal smoothness while generating embedding of consecutive snapshots by incorporating a temporal regularizer into a matrix factorization framework. A standard block-coordinate gradient descent (BCGD) algorithm is provided as a solution to the optimization problem. They also present two lemmas which prove the correctness of the method, followed by a thorough theoretical analysis of the solution. The time complexity of BCGD algorithm

is  $O(k \sum_T (n + m_T))$ , where  $n$  is the number of nodes,  $m_T$  is the number of edges in the graph  $G_T$ ,  $k$  is the number of dimensions, and  $T$  is the number of timestamps. In the later section, the authors describe two variants of the proposed algorithm, namely local BCGD algorithm and incremental BCGD algorithm, with local and incremental updates respectively. Then they compare the proposed methods with other latent space inferring approaches in terms of inference time and memory consumption, and prove the quality of learned latent spaces in terms of their link predictive power.

Another perspective of a dynamic network is a temporal network [50, 93], whose edges are active only when an interaction happens between the nodes. These interactions may lead to a flow of information between the nodes. Continuous-time dynamic network embedding (CTDNE) [92] aims at developing embedding for temporal networks by incorporating temporal dependencies into the state-of-the-art methods. In a temporal network, each edge is labeled with a timestamp which denotes the time of activation of an edge. CTDNE first perform a temporal random walk where an edge is traversed in the increasing order of timestamps and generates time-aware node sequences. Further, CTDNE uses the skip-gram architecture to learn time preserving node embeddings from the node sequences. The optimization objective of CTDNE is defined as

$$\max_f \log P_r(W_T = \{v_{i-w}, \dots, v_{i+w}\} \setminus v_i | f(v_i)) \tag{19}$$

where  $v_{i-w}, \dots, v_{i+w}$  is the neighboring vertices of vertex  $v_i$  and  $w$  is the context window size. The objective is to learn a function  $f$  which generates the node embeddings by maximizing the probability that the vertices co-occur in the temporal random walk occupy closer in the latent space. The time complexity of CTDNE is same as that of deepwalk as both uses the skip-gram architecture for learning node embeddings.

DynGEM [39] is a deep autoencoder based architecture to embed a dynamic network, which is inspired from SDNE. Given  $n$  snapshots of a dynamic network, DynGEM incrementally builds the embedding of the snapshot at time  $t_n$  from the embedding of the snapshot at time  $t_{n-1}$ . At each time step, DynGEM initializes the embedding from the previous time step, performs incremental learning, and thereby reduces the time for convergence from the second iteration. Incremental learning can be viewed as a transfer learning task where the model only needs to learn the changes between two graph snapshots. The dynamic network may grow in size. DynGEM uses a heuristic, prop size, to dynamically determine the number of hidden units required for each snapshot. The authors also provide various stability metrics for generating stable dynamic network embeddings. Unlike SDNE which uses

a sigmoid function for activations and SGD for training, DynGEM uses ReLU in all autoencoder layers to support weighted graphs, and Nesterov momentum with properly tuned hyperparameters for training. The loss function of DynGEM is stated as

$$L = \gamma' ||(Y - X) \odot B||^2 + \alpha' \sum_{i=1, j=1}^n a_{ij} ||\phi_i - \phi_j||_2^2 + \frac{\lambda}{2} \sum_{k=1}^K (||W^{(k)}||_F^2 + ||\hat{W}^{(k)}||_F^2) \quad (20)$$

where the first and second term represents the second order and first order proximities respectively. DynGEM uses similar optimization objective as that of SDNE [129], but unlike SDNE which operates on static network, DynGEM optimizes the parameters of the objective function at each time step, thereby by learn the parameters across a series of snapshots. The time complexity of DynGEM is  $O(ncdit)$ , where  $n$  is the number of vertices,  $d$  is the embedding dimension,  $c$  is the average degree of the network and  $i$  is the number of iterations, and  $t$  is the number of snapshots.

DynamicTriad [146] is another dynamic network embedding method which tries to preserve both structural and evolution pattern of the network. The aim of the work is to capture the network dynamics and to learn the low dimensional vectors for each node at different time steps. The work considers triadic closure as an important phenomenon which leads to network evolution, and is used to preserve the temporal dynamics while generating embedding. DynamicTriad models an optimization function with three components which is stated as

$$\sum_{t=1}^T (L_{sh,1}^t + \beta_0 L_{tr,2}^t) + \beta_1 \sum_{t=1}^{T-1} \sum_{i=1}^N ||u_i^{t+1} - u_i^t||_2^2 \quad (21)$$

where  $L_{sh,1}^t$  is a loss function to preserve the structural connectivity,  $L_{tr,2}^t$  is a loss function to preserve the triadic closure process and  $\sum_{t=1}^{T-1} \sum_{i=1}^N ||u_i^{t+1} - u_i^t||_2^2$  is a loss function to impose temporal smoothness by minimizing the euclidean distance between embedding vectors in adjacent time steps. DyRep [120] is another work which considers both topological evolution and temporal interactions, and aims to develop embeddings which encode both structural and temporal information.

## 5 Datasets for network representation learning

In this section, we discuss the major network datasets used in network representation learning research.

### 5.1 Social networks

*BlogCatalog* [117] This is a dataset used in most of the network representation learning research. BlogCatalog is a social network denoting the relationship between blogger authors listed on BlogCatalog website. The topic category of each author can act as the label of each node. To model BlogCatalog as an attributed network, we may use tags and short description of blogs as user attributes. Blogger users and groups can be considered as heterogeneous node types, and can form a heterogeneous network with user-user and user-group edges.

*Yelp*<sup>2</sup> This network represents the user friendship relationship in the Yelp social networking service. User reviews can be considered as the attribute information. A heterogeneous network can be modeled using users (U), businesses (B), cities (C) and categories (T) as nodes, and friendships (U-U), user reviews (B-U), business cities (B-C), and business categories (B-T) as edges.

*Flickr* [117] The Flickr network denotes contacts between users of photo sharing website Flickr. The interest group of each user can be used as the label of each node. To model Flickr as an attributed network, we may use aggregate tags on the user photos as user attributes.

*Youtube* [118] It is a social network where users are linked if they share a common video. The users can be grouped based on their tastes and can form the label of each user.

*Facebook Twitter* [65] Social networks showing friend and following relationships between users. They are usually used in works which use networks with the scale-free property.

### 5.2 Citation networks

*DBLP* [113] Three datasets using DBLP data (author citation network, paper citation network and co-authorship network) are used in NRL research. Author citation network links authors when one author cites the other, paper citation network links papers when one paper cites the other and co-authorship network links authors if they co-author at least a single paper. Paper title or paper abstract can be used as the attribute associated with each node.

*ArXiv* [66] Two datasets using ArXiv, ArXiv-GR-QC and ArXiv Astro-PH are used in network representation learning research. Both are co-author collaboration networks where authors are linked if they co-author at least a single paper.

*Citeseer<sup>3</sup> and Cora* [83] Data from both Citeseer and Cora is used as paper citation network with paper text denoting the attributes of nodes.

<sup>2</sup> [https://www.yelp.com/dataset\\_challenge](https://www.yelp.com/dataset_challenge).

<sup>3</sup> <http://konect.uni-koblenz.de/networks/citeseer>.

*Aminer computer science data* [113] and *database information system data* [112] These datasets are commonly used to model heterogeneous networks with the author, paper, and venue as node types and with author–author, author–paper, and paper–venue edges.

### 5.3 Other networks

*Wikipedia* [77] A language network using word co-occurrences can be constructed from Wikipedia data with POS tags as node labels. Wikieditor [140] is a signed network extracted from Wikipedia dataset [61]. Positive or negative edges are given based on the co-edit relationship between the users.

*PPI* [11] Protein–Protein Interaction Network(PPI) is a biological network showing the interaction between proteins. Protein functions or post-translational modifications can be considered as node labels.

*Epinions*<sup>4</sup> and *Slashdot*<sup>5</sup> Epinions is a user-user signed network constructed from product review site Epinions. Positive and negative links between users indicate the trust and distrust between them. The product review written by the users can be considered as the attributes. Slashdot is a technology news site which allows users to annotate other users as friends.

*Dynamic network datasets* Collaboration network snapshots from HEP-TH dataset [36], autonomous system communication network snapshots [64] from BGP (Border Gateway Protocol) logs, email communication network snapshots from ENRON dataset [58], user collaboration network snapshots from Github data,<sup>6</sup> timestamped communication networks from Chinese Telecom and PPDai [146], academic network snapshots from Aminer dataset<sup>7</sup> etc. are some dynamic network snapshots that are used to conduct the experiments with representation learning on dynamic networks.

## 6 Applications of network representation learning

Researchers applied network representation learning on various network mining applications and demonstrated the performance improvement of such tasks over the state-of-the-art methods. A pipeline of network embedding based network mining is shown in Fig. 10. In this

section, we discuss the major applications of network representation learning.

### 6.1 Node classification

Node classification [8] is the process of assigning labels to the unlabeled nodes in a network by considering the labels assigned to the labeled nodes and the topological structure of the network. The task is classified into single-label and multi-label node classification [59] depending upon the number of labels to be assigned to each node. A network embedding approach for node classification can be explained in three steps. (1) Embed the network to a low dimensional space. (2) Associate the known labels with the nodes, which form the training set (3) A lib-linear [30] classifier is trained to build the model, and can be used to predict the label of unlabeled nodes. The efficiency of the task can be measured using several evaluation measures like micro-F1, macro-F1 and accuracy. Node classification has been widely used as a benchmark for testing the efficiency of network representation methods. The effect of network embedding on node classification was tested on different datasets by various methods discussed in section 4.1 and the results presented by the authors are summarized below.

DeepWalk used social networks(BlogCatalog, Flickr, Youtube), Node2vec used social, biological, and language networks(BlogCatalog, P2P, Wikipedia), LINE used social, citation, and language networks (Flickr, Youtube, DBLP, Wikipedia), SDNE used social networks(BlogCatalog, Flickr, Youtube), GCN used citation networks(Citeseer, Cora, PubMed), HARP used social and citation networks(BlogCatalog, DBLP, Citeseer), ANE used citation and language networks(Cora, Citeseer, Wikipedia), GraphGAN used social and language networks(BlogCatalog, Wikipedia), and NETMF used social, biological, and language networks(BlogCatalog, Flickr, PPI, Wikipedia), for conducting the experiments with node classification problem on homogeneous networks. The effect of network embedding on node classification has been tested on attributed networks by TADW using citation and language networks(Cora, Citeseer, Wikipedia), AANE using social networks(BlogCatalog, Flickr, Youtube), DANE using social and citation networks (BlogCatalog, Flickr, DBLP, Epinions), IIRL using social and citation networks (BlogCatalog, Flickr, DBLP), and TriDNR using citation networks (Citeseer, DBLP). Experiment on node classification was tested on heterogeneous networks by Metapath2vec using citation networks (Aminer), and HIN2Vec using social and citation networks (BlogCatalog, Yelp, DBLP). The effect of network embedding on node classification has been tested on signed networks by SNE using language network(Wikieditor), SiNE using social networks( Epinions,

<sup>4</sup> <http://www.epinions.com/>.

<sup>5</sup> <https://slashdot.org/>.

<sup>6</sup> <https://www.gharchive.org/>.

<sup>7</sup> <http://www.aminer.org>.



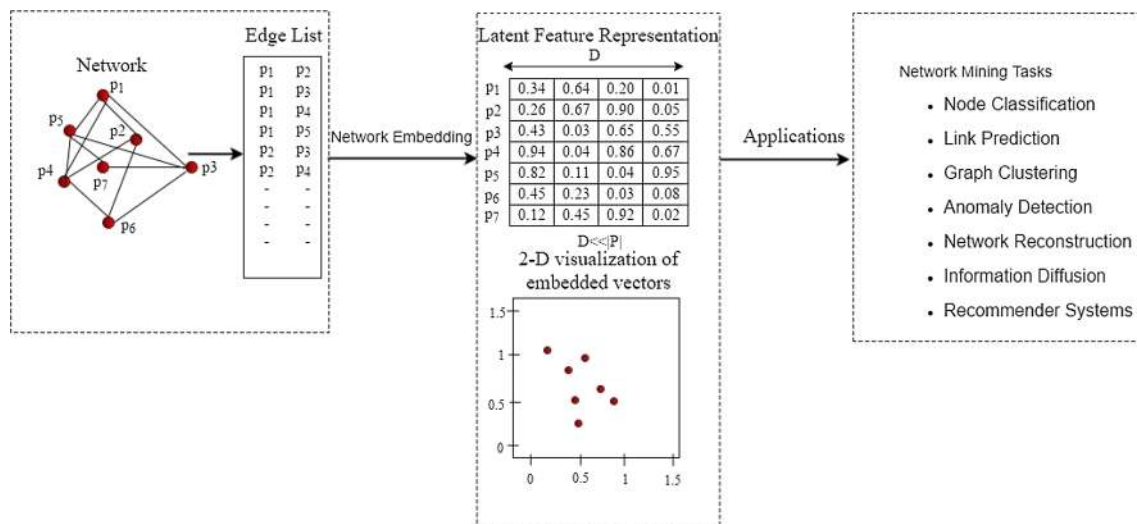


Fig. 10 A pipeline of network embedding based network mining

Slashdot), and SIDE using social and language networks (Epinions, Slashdot, Wikipedia) and on dynamic networks by DynamicTriad using communication and citation networks (Mobile, Loan, Aminer).

## 6.2 Link prediction

Link prediction [71, 73] is the one among the most well-studied network mining tasks that has got greater attention in recent years due to its wide range of applications. The link prediction problem can be defined as, given a social network at time  $t_1$ , the model needs to predict the edges that will be added to the network during the interval from current time  $t_1$  to a given future time  $t_2$ . In general, it can be related to the problem of inferring missing links from an observed network. Link prediction is useful in a variety of domains such as in social networks, where it recommends real-world friends, and in genomics, where it discovers the novel interaction between genes. The traditional method for link prediction is to define a similarity score between nodes based on similarity measures [71] like common neighbors, Adamic-Adar and preferential attachment. In a network embedding approach for link prediction, the nodes are first mapped into a low dimensional space. Then the vector similarity measures like cosine similarity and nearest neighbor approximation can be used to score the predicted links. The efficiency of the link prediction task can be measured using several evaluation measures such as precision and area under receiver operating curve (AOC) [73].

Node2vec used social, biological, and citation networks (Facebook, PPI, ArXiv), and SDNE and GraphGAN used citation networks (ArXiv) for conducting link prediction

experiments on homogeneous networks. Link prediction experiments were conducted on attributed networks by DeepGL using various network datasets available at network repository,<sup>8</sup> and on heterogeneous networks by HIN-2Vec using social and citation networks (BlogCatalog, Yelp, DBLP). The effect of network embedding on link prediction has been studied in signed networks by SNE using social and language networks (Slashdot, Wikieditor), SiNE using social networks (Epinions, Slashdot), SIDE using social and language networks (Epinions, Slashdot, Wiki), and SNEA using social networks (Epinions, Slashdot). Link prediction is an important challenge in dynamic networks and the significance of using node representations for link prediction in dynamic networks was tested by TNE using various network datasets from Koblenz Large Network Collection,<sup>9</sup> DynamicTriad using communication and citation networks (Mobile, Loan, Academic), DynGem using communication and citation networks (HEP-TH, ENRON, AS), CTDNE using various temporal network datasets, and DyRep using Github social network snapshots.

## 6.3 Network visualization

A network can be meaningfully visualized by creating a layout in 2-D space. In a network embedding approach for visualization, the learned node embeddings generated by the embedding algorithm is passed to a visualization tool (t-SNE [75], tensor flow embedding projector [1], PCA plot), and is visualized in a two-dimensional vector space.

<sup>8</sup> <http://networkrepository.com/>.

<sup>9</sup> <http://konect.uni-koblenz.de/>.



The visualization of the same dataset may differ across different embedding algorithms due to the differences in the properties preserved by each method.

Homogeneous networks are visualized by DNGR using t-SNE visualization of Wine Dataset<sup>10</sup> and ANE using t-SNE visualization of paper citation network (DBLP). TriDNR gives t-SNE visualization of attributed citation network (Citeseer). Metapath2vec provides tensorflow embedding projector visualization of a heterogeneous network (Aminer). SNE provides t-SNE visualization of a signed network (Wikieditor). Dynamic network snapshots are visualized by DynGem using a synthetic network (SYS), and DynRep using user collaboration network (Github).

## 6.4 Node clustering

Node clustering [105] is the process of grouping the nodes in a network into different clusters such that the sparsely connected dense subgraphs will be separated from each other. Functional module identification [28] in PPI networks is a typical application of node clustering. Traditional approaches for graph clustering [105] include methods based on k-spanning tree, betweenness centrality, shared nearest neighbor and clique enumeration. In a network embedding based approach for node clustering, the nodes are first mapped to a low dimensional space and vector space based clustering methods (eg. K-means clustering) are applied to generate the node clusters. Accuracy (AC) and normalized mutual information (NMI) [110] are the commonly used measures for evaluating the performance of node clustering task.

Some works used node clustering as the benchmark for evaluating the quality of node embeddings. DNGR performed node clustering on homogeneous language network (20-newsgroup network), DANE performed node clustering on attributed social and citation networks (BlogCatalog, Flickr, Epinions, DBLP), Methapath2vec and HNE performed node clustering on heterogeneous citation and social networks (Aminer, BlogCatalog), and SNEA performed node clustering on signed social networks (Epinions, Slashdot).

## 6.5 Other applications

Network representation learning is also applied in other areas of data mining and information retrieval. SDNE, HNE, and DynGem used network embedding for network reconstruction. GraphGAN used network embedding to build a recommender system using Movielens dataset. A user-movie bipartite graph is constructed and used the

learned representations of users and movies to recommend unwatched movies to the user. CUNE [143] aimed at enhancing the recommender system by incorporating the social information from the user-item bipartite network with rating information. CUNE constructs a user-interaction network from the user-item bipartite network, extracts implicit social information by embedding nodes of the user interaction network, and finally learns an objective function that incorporates top-k social links with the matrix factorization framework. Researchers [20] used network embedding (DeepWalk) to analyze Wikipedia pages for identifying historical analogies. The work [54] aimed at predicting users multi-interests from user interactions on health-related datasets. Other applications of network embedding include anomaly detection [39], multimodal search [16], information diffusion [10, 145], community detection [145], anchor-link prediction [80], emerging relation detection [144], sentiment link prediction [131], author identification [19], social relation extraction [123], and name disambiguation [142].

## 7 Conclusion and future works

As revolutionary advances in representation learning have got tremendous success in several application domains, the area of network mining also got influenced by representation learning techniques, due to its high-quality result and state-of-the-art performance. Various approaches based on representation learning were developed to learn node representations from large and complex networks. In this paper, we build a taxonomy of network representation learning methods based on the type of networks and review the major research works that come under each category. We further discuss the various network datasets used in network representation learning research. Finally, we review the major applications of network embedding.

Network representation learning is a young and promising field with a lot of unsolved challenges which provides various directions for future works.

*Preserving complex structure and properties of real-world networks:* Most of the real-world networks are very complex, and may contain higher order structures like network motifs [87]. They also exhibit complex properties, which include scale-free property, hyper edges, and nodes with high betweenness centrality. Even if some efforts have been made to work with scale-free property [31] and hyper networks [43, 124], a significant improvement has to be made in these directions.

*Complex network types:* The taxonomy of the types of networks that we provided in this review is not mutually exclusive. More complex network types can be modeled

<sup>10</sup> <https://archive.ics.uci.edu/ml/datasets/wine>.

by combining these basic types. For example, a citation network can be modeled as a dynamic heterogeneous attributed network which demands novel efforts in generating node embeddings.

*Addressing the big graph challenge:* Many real-world networks are very large with millions of nodes and vertices. Even if most of the embedding methods are designed to be highly scalable, a significant amount of work is to be done to adapt them towards such huge networks. As network embedding is basically an optimization problem, large-scale optimization methods can be used to improve its scalability. Another interesting direction towards enhancing the scalability is to develop new embedding strategies which can make use of the large-scale graph processing platforms like Giraph and Graphx, or to parallelize the existing methods so as to work with these distributed computing platforms.

*More Applications:* Most of the research on network embedding focused on node classification, node clustering, and link prediction. Network mining is a fast growing field with a lot of applications in various domains. So there exists an exciting direction of further work towards extending the existing methods or developing novel embedding methods towards solving more network mining tasks such as network evolution detection [135, 136], influential node detection [79], and network summarization [72].

A few efforts are already made to learn hyperbolic embeddings [25] and to use deep reinforcement learning [100] for network embedding, and more work is to be done in these significant directions.

**Acknowledgements** The authors would like to thank the management and staff of Department of Computer Applications, CUSAT, India and NSS College of Engineering, Palakkad, India for providing enough materials to conduct this study.

## Compliance with ethical standards

**Conflict of Interest** The authors declare that they have no conflict of interest.

## References

- Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, Devin M, Ghemawat S, Irving G, Isard M et al (2016) Tensorflow: a system for large-scale machine learning. *OSDI* 16:265–283
- Ahmed A, Shervashidze N, Narayanamurthy S, Josifovski V, Smola AJ (2013) Distributed large-scale natural graph factorization. In: Proceedings of the 22nd international conference on World Wide Web. ACM, pp 37–48
- Aioli F, Donini M, Navarin N, Sperduti A (2015) Multiple graph-kernel learning. In: 2015 IEEE symposium series on computational intelligence. IEEE, pp 1607–1614
- Barabási AL, Gulbahce N, Loscalzo J (2011) Network medicine: a network-based approach to human disease. *Nat Rev Genet* 12(1):56
- Bastian M, Heymann S, Jacomy M et al (2009) Gephi: an open source software for exploring and manipulating networks. *ICWSM* 8:361–362
- Belkin M, Niyogi P (2002) Laplacian eigenmaps and spectral techniques for embedding and clustering. In: Advances in neural information processing systems. pp 585–591
- Bengio Y, Courville A, Vincent P (2013) Representation learning: a review and new perspectives. *IEEE Trans Pattern Anal Mach Intell* 35(8):1798–1828
- Bhagat S, Cormode G, Muthukrishnan S (2011) Node classification in social networks. In: Aggarwal C (ed) Social network data analytics. Springer, pp 115–148
- Bottou L (2010) Large-scale machine learning with stochastic gradient descent. In: Proceedings of COMPSTAT'2010. Springer, pp 177–186
- Bourgault S, Lagnier C, Lamprier S, Denoyer L, Gallinari P (2014) Learning social network embeddings for predicting information diffusion. In: Proceedings of the 7th ACM international conference on Web search and data mining. ACM, pp 393–402
- Breitkreutz BJ, Stark C, Regulj T, Boucher L, Breitkreutz A, Livstone M, Oughtred R, Lackner DH, Bähler J, Wood V et al (2007) The biogrid interaction database: 2008 update. *Nucleic Acids Res suppl* 1(36):D637–D640
- Bullinaria JA, Levy JP (2007) Extracting semantic representations from word co-occurrence statistics: a computational study. *Behav Res Methods* 39(3):510–526
- Cai H, Zheng VW, Chang K (2018) A comprehensive survey of graph embedding: problems, techniques and applications. *IEEE Trans Knowl Data Eng* 30(9):1616–1637
- Cao S, Lu W, Xu Q (2015) Grarep: Learning graph representations with global structural information. In: Proceedings of the 24th ACM international on conference on information and knowledge management. ACM, pp 891–900
- Cao S, Lu W, Xu Q (2016) Deep neural networks for learning graph representations. In: AAAI. pp 1145–1152
- Chang S, Han W, Tang J, Qi GJ, Aggarwal CC, Huang TS (2015) Heterogeneous network embedding via deep architectures. In: Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 119–128
- Chen H, Perozzi B, Hu Y, Skiena S (2017a) Harp: Hierarchical representation learning for networks. *arXiv preprint arXiv:1706.07845*
- Chen J, Ma T, Xiao C (2018) Fastgcn: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247*
- Chen T, Sun Y (2017) Task-guided and path-augmented heterogeneous network embedding for author identification. In: Proceedings of the Tenth ACM international conference on web search and data mining. ACM, pp 295–304
- Chen Y, Perozzi B, Skiena S (2017b) Vector-based similarity measurements for historical figures. *Inf Syst* 64:163–174
- Collobert R, Weston J (2008) A unified architecture for natural language processing: deep neural networks with multitask learning. In: Proceedings of the 25th international conference on machine learning. ACM, pp 160–167
- Cui P, Wang X, Pei J, Zhu W (2017) A survey on network embedding. *arXiv preprint arXiv:1711.08752*
- Dahl GE, Yu D, Deng L, Acero A (2012) Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Trans Audio Apeech Lang Process* 20(1):30–42
- Dai Q, Li Q, Tang J, Wang D (2017) Adversarial network embedding. *arXiv preprint arXiv:1711.07838*

25. De Sa C, Gu A, Ré C, Sala F (2018) Representation tradeoffs for hyperbolic embeddings. arXiv preprint [arXiv:180403329](https://arxiv.org/abs/180403329)
26. Dean J, Ghemawat S (2008) Mapreduce: simplified data processing on large clusters. *Commun ACM* 51(1):107–113
27. Defferrard M, Bresson X, Vandergheynst P (2016) Convolutional neural networks on graphs with fast localized spectral filtering. In: *Advances in neural information processing systems*. pp 3844–3852
28. Dittrich MT, Klau GW, Rosenwald A, Dandekar T, Müller T (2008) Identifying functional modules in protein-protein interaction networks: an integrated exact approach. *Bioinformatics* 24(13):i223–i231
29. Dong Y, Chawla NV, Swami A (2017) metapath2vec: scalable representation learning for heterogeneous networks. In: *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, pp 135–144
30. Fan RE, Chang KW, Hsieh CJ, Wang XR, Lin CJ (2008) Liblinear: a library for large linear classification. *J Mach Learn Res* 9(Aug):1871–1874
31. Feng R, Yang Y, Hu W, Wu F, Zhuang Y (2017) Representation learning for scale-free networks. arXiv preprint [arXiv:1711.0755](https://arxiv.org/abs/1711.0755)
32. Fu Ty, Lee WC, Lei Z (2017) Hin2vec: explore meta-paths in heterogeneous information networks for representation learning. In: *Proceedings of the 2017 ACM on conference on information and knowledge management*. ACM, pp 1797–1806
33. Fu Y, Ma Y (2012) *Graph embedding for pattern analysis*. Springer, Berlin
34. Gallagher B, Eliassi-Rad T (2010) Leveraging label-independent features for classification in sparsely labeled networks: an empirical study. In: *Advances in social network mining and analysis*. Springer, pp 1–19
35. García-Durán A, Niepert M (2017) Learning graph representations with embedding propagation. arXiv preprint [arXiv:1710.3059](https://arxiv.org/abs/1710.3059)
36. Gehrke J, Ginsparg P, Kleinberg J (2003) Overview of the 2003 kdd cup. *ACM SIGKDD Explor Newsl* 5(2):149–151
37. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial nets. In: *Advances in neural information processing systems*. pp 2672–2680
38. Goyal P, Ferrara E (2017) Graph embedding techniques, applications, and performance: a survey. arXiv preprint [arXiv:1705.2801](https://arxiv.org/abs/1705.2801)
39. Goyal P, Kamra N, He X, Liu Y (2018) Dyngem: deep embedding method for dynamic graphs. arXiv preprint [arXiv:1805.11273](https://arxiv.org/abs/1805.11273)
40. Graves A, Mohamed Ar, Hinton G (2013) Speech recognition with deep recurrent neural networks. In: *2013 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, pp 6645–6649
41. Greff K, Srivastava RK, Koutník J, Steunebrink BR, Schmidhuber J (2017) Lstm: a search space odyssey. *IEEE Trans Neural Netw Learn Syst* 28(10):2222–2232
42. Grover A, Leskovec J (2016) node2vec: Scalable feature learning for networks. In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp 855–864
43. Gui H, Liu J, Tao F, Jiang M, Norick B, Han J (2016) Large-scale embedding learning in heterogeneous event data. In: *2016 IEEE 16th international conference on data mining (ICDM)*. IEEE, pp 907–912
44. Guo Y, Liu Y, Oerlemans A, Lao S, Wu S, Lew MS (2016) Deep learning for visual understanding: a review. *Neurocomputing* 187:27–48
45. Hamilton W, Ying Z, Leskovec J (2017a) Inductive representation learning on large graphs. In: *Advances in neural information processing systems*. pp 1025–1035
46. Hamilton WL, Ying R, Leskovec J (2017b) Representation learning on graphs: methods and applications. arXiv preprint [arXiv:170905584](https://arxiv.org/abs/170905584)
47. Hammond DK, Vandergheynst P, Gribonval R (2011) Wavelets on graphs via spectral graph theory. *Appl Comput Harmonic Anal* 30(2):129–150
48. Hinton G, Deng L, Yu D, Dahl GE, Ar Mohamed, Jaitly N, Senior A, Vanhoucke V, Nguyen P, Sainath TN et al (2012) Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. *IEEE Signal Process Mag* 29(6):82–97
49. Hinton GE, Osindero S, Teh YW (2006) A fast learning algorithm for deep belief nets. *Neural comput* 18(7):1527–1554
50. Holme P, Saramäki J (2012) Temporal networks. *Phys Rep* 519(3):97–125
51. Huang X, Li J, Hu X (2017a) Accelerated attributed network embedding. In: *Proceedings of the 2017 SIAM international conference on data mining*. SIAM, pp 633–641
52. Huang X, Li J, Hu X (2017b) Label informed attributed network embedding. In: *Proceedings of the tenth ACM international conference on web search and data mining*. ACM, pp 731–739
53. Huang X, Song Q, Li J, Hu X (2018) Exploring expert cognition for attributed network embedding. In: *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, pp 270–278
54. Jin Z, Liu R, Li Q, Zeng DD, Zhan Y, Wang L (2016) Predicting user's multi-interests with network embedding in health-related topics. In: *2016 International joint conference on neural networks (IJCNN)*. IEEE, pp 2568–2575
55. Kim J, Park H, Lee JE, Kang U (2018) Side: Representation learning in signed directed networks. In: *Proceedings of the 2018 World Wide Web conference on World Wide Web, international World Wide Web conferences steering committee*. pp 509–518
56. Kipf TN, Welling M (2016a) Semi-supervised classification with graph convolutional networks. arXiv preprint [arXiv:160902907](https://arxiv.org/abs/160902907)
57. Kipf TN, Welling M (2016b) Variational graph auto-encoders. arXiv preprint [arXiv:161107308](https://arxiv.org/abs/161107308)
58. Klimt B, Yang Y (2004) The enron corpus: a new dataset for email classification research. In: *European conference on machine learning*. Springer, pp 217–226
59. Kong X, Shi X, Yu PS (2011) Multi-label collective classification. In: *Proceedings of the 2011 SIAM international conference on data mining*. SIAM, pp 618–629
60. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems*. pp 1097–1105
61. Kumar S, Spezzano F, Subrahmanian V (2015) Vews: A wikipedia vandal early warning system. In: *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, pp 607–616
62. Le Q, Mikolov T (2014) Distributed representations of sentences and documents. In: *International conference on machine learning*. pp 1188–1196
63. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. *Nature* 521(7553):436
64. Leskovec J, Krevl A (2014) SNAP datasets: Stanford large network dataset collection. <http://snap.stanford.edu/data>
65. Leskovec J, McAuley JJ (2012) Learning to discover social circles in ego networks. In: *Advances in neural information processing systems*. pp 539–547
66. Leskovec J, Kleinberg J, Faloutsos C (2007) Graph evolution: densification and shrinking diameters. *ACM Trans Knowl Discov Data (TKDD)* 1(1):2

67. Leskovec J, Huttenlocher D, Kleinberg J (2010) Signed networks in social media. In: Proceedings of the SIGCHI conference on human factors in computing systems. ACM, pp 1361–1370
68. Levy O, Goldberg Y, Dagan I (2015) Improving distributional similarity with lessons learned from word embeddings. *Trans Assoc Comput Linguist* 3:211–225
69. Li J, Zhu J, Zhang B (2016) Discriminative deep random walk for network classification. In: Proceedings of the 54th annual meeting of the association for computational linguistics (Volume 1: Long Papers). vol 1, pp 1004–1013
70. Li J, Dani H, Hu X, Tang J, Chang Y, Liu H (2017) Attributed network embedding for learning in a dynamic environment. In: Proceedings of the 2017 ACM on conference on information and knowledge management. ACM, pp 387–396
71. Liben-Nowell D, Kleinberg J (2007) The link-prediction problem for social networks. *J Assoc Inf Sci Technol* 58(7):1019–1031
72. Liu Y, Safavi T, Dighe A, Koutra D (2016) Graph summarization methods and applications: a survey. arXiv preprint [arXiv:161204883](https://arxiv.org/abs/161204883)
73. Lü L, Zhou T (2011) Link prediction in complex networks: a survey. *Physica A Stat Mech Appl* 390(6):1150–1170
74. Lü L, Medo M, Yeung CH, Zhang YC, Zhang ZK, Zhou T (2012) Recommender systems. *Phys Rep* 519(1):1–49
75. Lvd Maaten, Hinton G (2008) Visualizing data using t-SNE. *J Mach Learn Res* 9(Nov):2579–2605
76. Maayan A (2011) Introduction to network analysis in systems biology. *Sci Signal* 4(190):tr5
77. Mahoney M (2011) Large text compression benchmark. <http://www.mattmahoney.net/text/text.html>
78. Malewicz G, Austern MH, Bik AJ, Dehnert JC, Horn I, Leiser N, Czajkowski G (2010) Pregel: a system for large-scale graph processing. In: Proceedings of the 2010 ACM SIGMOD international conference on management of data. ACM, pp 135–146
79. Malliaros FD, Rossi MEG, Vazirgiannis M (2016) Locating influential nodes in complex networks. *Sci Rep* 6:19307
80. Man T, Shen H, Liu S, Jin X, Cheng X (2016) Predict anchor links across social networks via an embedding approach. *IJCAI* 16:1823–1829
81. Martella C, Shaposhnik R, Logothetis D, Harenberg S (2015) Practical graph analytics with apache giraph. Springer, Berlin
82. Mason W, Vaughan JW, Wallach H (2014) Computational social science and social computing. *Mach learn* 95:257–260. <https://doi.org/10.1007/s10994-013-5426-8>
83. McCallum AK, Nigam K, Rennie J, Seymore K (2000) Automating the construction of internet portals with machine learning. *Inf Retr* 3(2):127–163
84. Micheli A (2009) Neural network for graphs: a contextual constructive approach. *IEEE Trans Neural Netw* 20(3):498–511
85. Mikolov T, Chen K, Corrado G, Dean J (2013a) Efficient estimation of word representations in vector space. arXiv preprint [arXiv:13013781](https://arxiv.org/abs/13013781)
86. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013b) Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. pp 3111–3119
87. Milo R, Shen-Orr S, Itzkovitz S, Kashtan N, Chklovskii D, Alon U (2002) Network motifs: simple building blocks of complex networks. *Science* 298(5594):824–827
88. Mnih A, Kavukcuoglu K (2013) Learning word embeddings efficiently with noise-contrastive estimation. In: Advances in neural information processing systems. pp 2265–2273
89. Moyano LG (2017) Learning network representations. *Eur Phys J Spec Top* 226(3):499–518
90. Natarajan N, Dhillon IS (2014) Inductive matrix completion for predicting gene-disease associations. *Bioinformatics* 30(12):i60–i68
91. Neville J, Jensen D (2000) Iterative classification in relational data. In: Proc. AAAI-2000 workshop on learning statistical models from relational data. pp 13–20
92. Nguyen GH, Lee JB, Rossi RA, Ahmed NK, Koh E, Kim S (2018) Continuous-time dynamic network embeddings. In: 3rd International workshop on learning representations for big networks (WWW BigNet)
93. Nicosia V, Tang J, Mascolo C, Musolesi M, Russo G, Latora V (2013) Graph metrics for temporal networks. In: Holme P, Sara-mäki J (eds) Temporal networks. Springer, Berlin, pp 15–40. [https://doi.org/10.1007/978-3-642-36461-7\\_2](https://doi.org/10.1007/978-3-642-36461-7_2)
94. Ou M, Cui P, Pei J, Zhang Z, Zhu W (2016) Asymmetric transitivity preserving graph embedding. In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp 1105–1114
95. Pan S, Wu J, Zhu X, Zhang C, Wang Y (2016) Tri-party deep network representation. *Network* 11(9):12
96. Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp 701–710
97. Perozzi B, Kulkarni V, Skiena S (2016) Walklets: Multiscale graph embeddings for interpretable network classification. arXiv preprint [arXiv:160502115](https://arxiv.org/abs/160502115)
98. Pfeiffer III JJ, Moreno S, La Fond T, Neville J, Gallagher B (2014) Attributed graph models: modeling network structure with correlated attributes. In: Proceedings of the 23rd international conference on World Wide Web. ACM, pp 831–842
99. Qi GJ, Aggarwal C, Tian Q, Ji H, Huang T (2012) Exploring context and content links in social media: a latent space method. *IEEE Trans Pattern Anal Mach Intell* 34(5):850–862
100. Qu M, Tang J, Han J (2018) Curriculum learning for heterogeneous star network embedding via deep reinforcement learning. In: Proceedings of the eleventh ACM international conference on web search and data mining. ACM, pp 468–476
101. Ribeiro LF, Saverese PH, Figueiredo DR (2017) struc2vec: Learning node representations from structural identity. In: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 385–394
102. Rossi RA, Zhou R, Ahmed NK (2017) Deep feature learning for graphs. arXiv preprint [arXiv:170408829](https://arxiv.org/abs/170408829)
103. Roweis ST, Saul LK (2000) Nonlinear dimensionality reduction by locally linear embedding. *Science* 290(5500):2323–2326
104. Scarselli F, Gori M, Tsoi AC, Hagenbuchner M, Monfardini G (2008) The graph neural network model. *IEEE Trans Neural Netw* 20(1):61–80
105. Schaeffer SE (2007) Graph clustering. *Comput Sci Rev* 1(1):27–64
106. Schmidhuber J (2015) Deep learning in neural networks: an overview. *Neural Netw* 61:85–117
107. Shaw B, Jebara T (2009) Structure preserving embedding. In: Proceedings of the 26th annual international conference on machine learning. ACM, pp 937–944
108. Socher R, Bengio Y, Manning CD (2012) Deep learning for nlp (without magic). In: Tutorial abstracts of ACL 2012, association for computational linguistics. pp 5–5
109. Steglich C, Snijders TA, Pearson M (2010) Dynamic networks and behavior: separating selection from influence. *Sociol Methodol* 40(1):329–393
110. Strehl A, Ghosh J, Mooney R (2000) Impact of similarity measures on web-page clustering. In: Workshop on artificial intelligence for web search (AAAI 2000). vol 58, p 64
111. Sun Y, Han J (2012) Mining heterogeneous information networks: principles and methodologies. *Synth Lect Data Min Knowl Discov* 3(2):1–159



112. Sun Y, Han J, Yan X, Yu PS, Wu T (2011) Pathsim: meta path-based top-k similarity search in heterogeneous information networks. *Proc VLDB Endow* 4(11):992–1003
113. Tang J, Zhang J, Yao L, Li J, Zhang L, Su Z (2008) Arnetminer: extraction and mining of academic social networks. In: *Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, pp 990–998
114. Tang J, Qu M, Mei Q (2015a) Pte: Predictive text embedding through large-scale heterogeneous text networks. In: *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, pp 1165–1174
115. Tang J, Qu M, Wang M, Zhang M, Yan J, Mei Q (2015b) Line: large-scale information network embedding. In: *Proceedings of the 24th international conference on World Wide Web, international World Wide Web conferences steering committee*. pp 1067–1077
116. Tang J, Chang Y, Aggarwal C, Liu H (2016) A survey of signed network mining in social media. *ACM Comput Surv (CSUR)* 49(3):42
117. Tang L, Liu H (2009a) Relational learning via latent social dimensions. In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp 817–826
118. Tang L, Liu H (2009b) Scalable learning of collective behavior based on sparse social dimensions. In: *Proceedings of the 18th ACM conference on Information and knowledge management*. ACM, pp 1107–1116
119. Tran DV, Navarin N, Sperduti A (2018) On filter size in graph convolutional networks. In: *2018 IEEE symposium series on computational intelligence (SSCI)*. IEEE, pp 1534–1541
120. Trivedi R, Farajtabar M, Biswal P, Zha H (2018) Representation learning over dynamic graphs. *arXiv preprint arXiv:180304051*
121. Tu C, Zhang W, Liu Z, Sun M (2016) Max-margin deepwalk: discriminative learning of network representation. In: *IJCAI*. pp 3889–3895
122. Tu C, Liu H, Liu Z, Sun M (2017a) Cane: context-aware network embedding for relation modeling. In: *Proceedings of the 55th annual meeting of the association for computational linguistics (Volume 1: Long Papers)*. vol 1, pp 1722–1731
123. Tu C, Zhang Z, Liu Z, Sun M (2017b) Transnet: translation-based network representation learning for social relation extraction. In: *Proceedings of international joint conference on artificial intelligence (IJCAI)*. Melbourne
124. Tu K, Cui P, Wang X, Wang F, Zhu W (2017c) Structural deep embedding for hyper-networks. *arXiv preprint arXiv:1711.0146*
125. Utsumi A (2015) A complex network approach to distributional semantic models. *PLoS ONE* 10(8):e0136277
126. Van Der Maaten L, Postma E, Van den Herik J (2009) Dimensionality reduction: a comparative. *J Mach Learn Res* 10:66–71
127. Vishwanathan SVN, Schraudolph NN, Kondor R, Borgwardt KM (2010) Graph kernels. *J Mach Learn Res* 11(Apr):1201–1242
128. Wan J, Wang D, Hoi SCH, Wu P, Zhu J, Zhang Y, Li J (2014) Deep learning for content-based image retrieval: a comprehensive study. In: *Proceedings of the 22nd ACM international conference on multimedia*. ACM, pp 157–166
129. Wang D, Cui P, Zhu W (2016) Structural deep network embedding. In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp 1225–1234
130. Wang H, Wang J, Wang J, Zhao M, Zhang W, Zhang F, Xie X, Guo M (2017a) Graphgan: Graph representation learning with generative adversarial nets. *arXiv preprint arXiv:171108267*
131. Wang H, Zhang F, Hou M, Xie X, Guo M, Liu Q (2018) Shine: signed heterogeneous information network embedding for sentiment link prediction. In: *Proceedings of the eleventh ACM international conference on web search and data mining*. ACM, pp 592–600
132. Wang S, Aggarwal C, Tang J, Liu H (2017b) Attributed signed network embedding. In: *Proceedings of the 2017 ACM on conference on information and knowledge management*. ACM, pp 137–146
133. Wang S, Tang J, Aggarwal C, Chang Y, Liu H (2017c) Signed network embedding in social media. In: *Proceedings of the 2017 SIAM international conference on data mining*. SIAM, pp 327–335
134. Wright J, Ma Y, Mairal J, Sapiro G, Huang TS, Yan S (2010) Sparse representation for computer vision and pattern recognition. *Proc IEEE* 98(6):1031–1044
135. Wu T, Chen L, Zhong L, Xian X (2017) Predicting the evolution of complex networks via similarity dynamics. *Physica A Stat Mech Appl* 465:662–672
136. Wu T, Chang CS, Liao W (2018) Tracking network evolution and their applications in structural network analysis. *IEEE Transactions on Network Science and Engineering*
137. Xin RS, Gonzalez JE, Franklin MJ, Stoica I (2013) Graphx: a resilient distributed graph system on spark. In: *First international workshop on graph data management experiences and systems*. ACM, p 2
138. Xu L, Wei X, Cao J, Philip SY (2018) On exploring semantic meanings of links for embedding social networks. *DEF* 2:6
139. Yang C, Liu Z, Zhao D, Sun M, Chang EY (2015) Network representation learning with rich text information. In: *IJCAI*. pp 2111–2117
140. Yuan S, Wu X, Xiang Y (2017) SNE: signed network embedding. In: *Pacific-Asia conference on knowledge discovery and data mining*. Springer, pp 183–195
141. Zampieri G, Van Tran D, Donini M, Navarin N, Aiolfi F, Sperduti A, Valle G (2018) Scuba: scalable kernel-based gene prioritization. *BMC Bioinform* 19(1):23
142. Zhang B, Al Hasan M (2017) Name disambiguation in anonymized graphs using network embedding. In: *Proceedings of the 2017 ACM on conference on information and knowledge management*. ACM, pp 1239–1248
143. Zhang C, Yu L, Wang Y, Shah C, Zhang X (2017a) Collaborative user network embedding for social recommender systems. In: *Proceedings of the 2017 SIAM international conference on data mining*. SIAM, pp 381–389
144. Zhang J, Lu CT, Zhou M, Xie S, Chang Y, Philip SY (2016) Heer: Heterogeneous graph embedding for emerging relation detection from news. In: *2016 IEEE international conference on big data (big data)*. IEEE, pp 803–812
145. Zhang J, Cui L, Fu Y (2017b) Latte: application oriented network embedding. *arXiv preprint arXiv:1711.11466*
146. Zhou L, Yang Y, Ren X, Wu F, Zhuang Y (2018) Dynamic network embedding by modeling triadic closure process. In: *Thirty-Second AAAI Conference on Artificial Intelligence*
147. Zhu L, Guo D, Yin J, Ver Steeg G, Galstyan A (2016) Scalable temporal latent space inference for link prediction in dynamic social networks. *IEEE Trans Knowl Data Eng* 28(10):2765–2777
148. Zhu W, Milanović JV (2017) Interdependency modeling of cyber-physical systems using a weighted complex network approach. In: *PowerTech, 2017 IEEE Manchester*. IEEE, pp 1–6
149. Zhu X, Ghahramani Z, Lafferty JD (2003) Semi-supervised learning using gaussian fields and harmonic functions. In: *Proceedings of the 20th international conference on machine learning (ICML-03)*. pp 912–919

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.