Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2019) 27: 4768 – 4782
© TÜBİTAK
doi:10.3906/elk-1807-333

Survey of network embedding techniques for social networks

Pranav NERURKAR*, **Madhav CHANDANE**, **Sunil BHIRUD**
Department of Computer Engineering and IT, Veermata Jijabai Technological Institute, University of Mumbai, Mumbai, India

Received: 30.07.2018 • **Accepted/Published Online:** 26.03.2019 • **Final Version:** 26.11.2019

Abstract: High dimensionality of data is a challenging scenario in the current era as the digital transformation of the society is in process. This problem is particularly complex in social networks as in such systems, it is coupled with other challenges such as interdependency of data points and heterogeneity of data sources. To overcome such disadvantages and aid in creation of downstream applications for social network analysis, network embedding techniques have been proposed. These techniques, in themselves, are not important but are the backbone of various network-based applications. Due to the scientific interest in this domain there has been a mushrooming of embedding techniques. It has therefore become crucial to learn the intuitions behind these techniques in order to compare and contrast them. The current analytical study is drawn with the following broad objectives: providing practitioners with understanding of network representative learning mathematical study of state-of-the-art techniques and highlighting the evolution of the literature in this field.

Key words: Dimensionality reduction, network embedding, latent space

1. Introduction

Representing actual or synthetic structures in the form of a social network (graph) has advantages associated with information processing [1, 2]. However, such a representation leads to interdependency among the various components of the data [2]. Due to such interdependency, it is not possible to design parallel or distributed architectures for processing the data [2]. Machine learning or deep learning frameworks assume that the data points are independent and identically distributed in a vector area. Hence, these techniques cannot be applied directly to social network data [3]. This is another disadvantage caused due to the ”vertex interdependency” in social networks.

A social network $X(t)$ has elements called as ‘covariates’ (side information or node attributes) that describe a node (vertex) [4, 5]. If a large number of covariates are present for each vertex the dimensionality of the data also becomes high (curse of dimensionality) [6–8].

The method used to address the issues of interdependency and high dimensionality in social networks is known as network representative learning (NRL). A NRL framework provides low-dimensional vector representations (embeddings) from high-dimensional data. These embeddings are representations of the vertices in a low-dimensional vector space. After the representations are learned, the inherent coupling between the nodes of the social network is no longer present and this makes the data amenable for further evaluation and downstream network applications viz. link prediction, clustering, node classification, and network visualization as shown in Figure 1.

*Correspondence: panerurkar_p16@ce.vjti.ac.in

Before network representation learning methods were proposed, social network analysis (SNA) was performed by means of extensive feature engineering. This required use of kernel functions or graph statistics making the entire exercise task-dependent. NRL frameworks eliminated the need for feature engineering and made SNA task-independent.

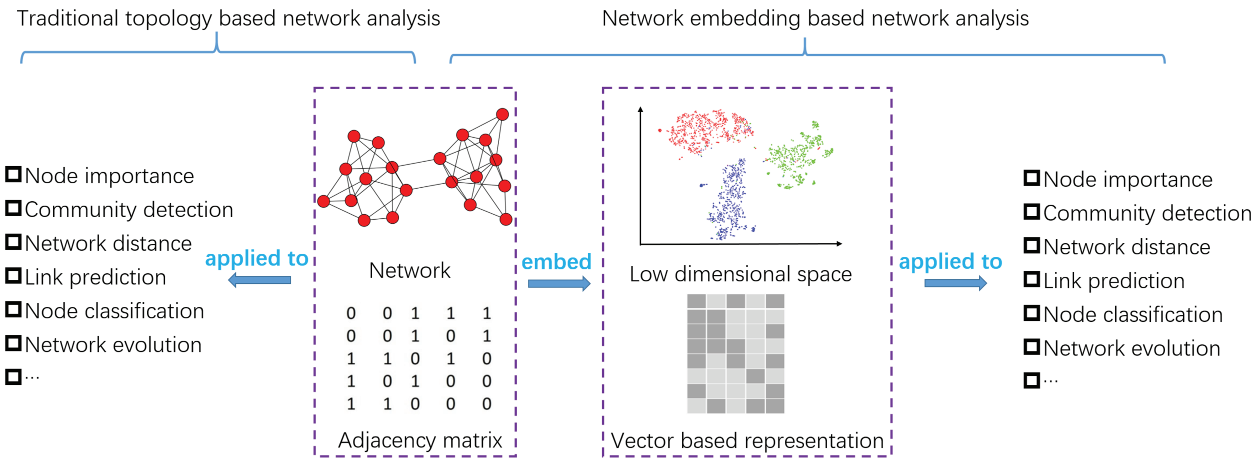

Figure 1. A comparison between feature-engineering-based network analysis and NRL-based network analysis. The image was extracted from Cui et al. [2]

The key challenge for NRL frameworks is preserving similarities of the original dimensions in the lower dimension space (latent dimensions). This is done by means of an encoder function that learns complex nonlinear relationships within the input data. These learned relationships are then used to represent (embed) the high dimension information into low dimension space. The advantage of NRL frameworks is that the produced graph embeddings do not have the aforementioned disadvantages that are present in social network data. The four classes of NRL techniques discussed in the literature are adjacency preserving methods [9–11, 13–18, 36], multihop distance preserving methods, neighborhood overlap preserving methods, and random walk occurrence preserving methods [19–28].

The intention of this paper is to provide a mathematical background and intuition behind the state-of-the-art network embedding models. It also describes the evolution of NRL frameworks from proximity preserving "shallow encoder" frameworks to deep learning frameworks [29–32]. Deep learning frameworks have achieved performance matching the state-of-the-art NRL techniques. The rest of the paper is organized as follows: Section 2 provides definitions and the problem statement of embedding techniques, a detailed taxonomy of the algorithms is given in Section 4, and the survey is concluded in Section 7.

2. Preliminaries

Given a graph $G = (V, E)$, the aim is to represent each node u in a low-dimensional vector space y_u by learning a mapping $f : V \rightarrow R^d$, namely $y_v = f(v) \forall v \in V$. It is required that $d << n$ and the function f preserve a proximity measure defined on the graph G. Intuitively, if two nodes u and v are "similar" in the original graph G, their embeddings y_u and y_v should be "close" to each other in the embedding space. Mathematically, "closeness" in the embedding space is denoted by dot product of the vectors i.e. $z_u^T z_v = 1$. However, there is no uniformity in defining the concept of "similarity" in the original graph.

3. Background

3.1. Goal of NRL frameworks

Given a social network $G(V,E)$ as shown in Figure 2 with dimensions $\mathbb{R}^{|V|*|V|*d}$, NRL frameworks learns a representation of it into a low dimension (latent) space $\mathbb{R}^{|V|*k}$ where $d <<< k$. The nodes of the original network are learned into real valued vector embeddings $\mathbb{R}^{k}$.

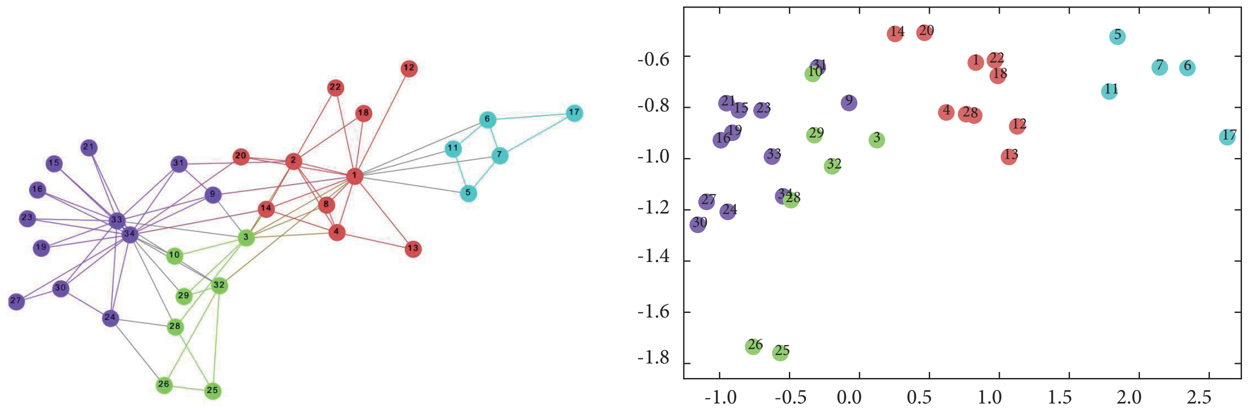

Figure 2. Zachary's Karate Club social network with communities. The original network represented in a low dimension space such that vector embeddings of the nodes in the same community in the original network are located close to each other in the low dimension space. The image was extracted from Perozzi et al. [19]

3.2. Components of NRL Frameworks

NRL frameworks consist of three components as given in Figure 3: An encoder (it converts nodes of the original social network into real valued vertex embeddings), a similarity measure in the original network (this similarity is preserved in latent space such that nodes that are "similar" in the original network are also "similar" in latent space) and an optimization process (this trains the parameters θ of the encoder to capture the similarity measure in the latent space).

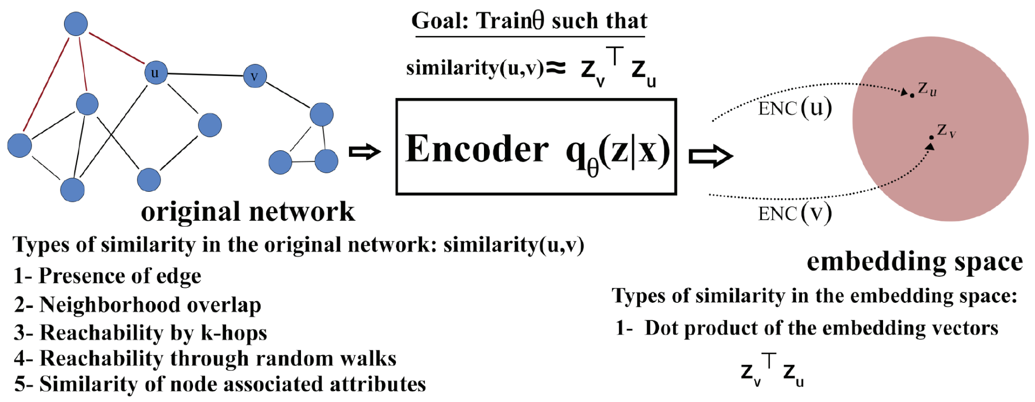
Goal: Trainθ such that
similarity(u,v)≈ $z_v^\top z_u$
u
v
Encoder $q_\theta(z|x)$
ENC(u)
ENC(v)
z_u
z_v
original network
Types of similarity in the original network: similarity(u,v)
1- Presence of edge
2- Neighborhood overlap
3- Reachability by k-hops
4- Reachability through random walks
5- Similarity of node associated attributes
embedding space
Types of similarity in the embedding space:
1- Dot product of the embedding vectors
$z_v^\top z_u$

Figure 3. Components of NRL Frameworks : The several classes of NRL techniques are due to differing definitions of the concept of similarity in the original space.

3.3. Types of NRL Frameworks

Adjacency preserving methods define similarity between two nodes u, v in the original network as the presence of an edge between them in the adjacency matrix A i.e. $A_{u,v} = 1$. Such nodes should have dot product of the vertex embeddings same as $z_v^T.z_u = 1$ when $A_{u,v} = 1$. The parameters θ of the encoder are trained by an optimization process such as Matrix factorization or gradient descent to capture this similarity and minimize the loss L as given in Eq. 1.

$$L = \sum_{(u,v)\in(V*V)} ||z_u^T z_v - A_{u,v}||^2. \tag{1}$$

Multihop distance preserving techniques define similarity in the original social network as the existence of a k-hop path between two nodes u, v i.e $A_{u,v}^k = 1$ where $A_{u,v}^k$ is the k-hop adjacency matrix. The optimization process trains the encoder to minimize the loss function given in Eq. 14. In effect, the encoder learns vertex embeddings to preserve the k-hop distance between the nodes.

$$L = \sum_{(u,v)\in(V*V)} ||z_u^T z_v - A_{u,v}^k||^2. \tag{2}$$

Neighborhood overlap preserving techniques define similarity in the original social network as the degree of overlap of neighborhoods N of two nodes $N_u \cap N_v$. Using an optimization process, the encoder is trained to minimize the loss function given in Eq. 3. Thus, the encoder learns vertex embeddings to preserve the neighborhood overlap matrix or second-order proximity between nodes.

$$L = \sum_{(u,v)\in(V*V)} ||z_u^T z_v - N_{u,v}||^2. \tag{3}$$

Random walk occurrence preserving techniques define similarity in the original social network as the probability of reaching a node v by a random walk from node u i.e. $P(v|u)$. The encoder learns vertex embeddings to preserve the random walk cooccurrence probability between the nodes as given by Eq. 4.

$$L = \sum_{(u\in V)} \sum_{(v\in N_r(u))} -log(P(v|z_u)). \tag{4}$$

4. Network embedding methods

4.1. Adjacency preserving techniques

Isomap determines the neighbors of each point. It constructs a neighborhood graph and uses it to compute the shortest path between two nodes. Nodes having shortest path are ”similar” in original high-dimension space. Isomap utilizes this nearest neighborhood similarity matrix for obtaining Z. The key problem of Isomap is its high complexity due to the computation of pair-wise shortest paths. Locally linear embedding (LLE) eliminates the need to estimate the pairwise distances between vertices. LLE transforms data into an affinity graph based on the feature vectors of nodes, before applying SGD to train embeddings [33]. Other adjacenc- based methods are landmark classical scaling [34], stochastic neighbor embedding (SNE), singular valued decomposition, and matrix factorization [13]. The key drawback of adjacency-based techniques is that they generate node embeddings that overfit the adjacency matrix of the original graph. This causes failure in detecting inconspicuous connections; hence, this category of techniques cannot be used for downstream network applications like link prediction. However, for applications such as graph reconstruction, adjacency-based methods are preferred. Other drawbacks are $O(|V|^2)$ run-time ($|V|^2$ node pairs have to be considered), $O(|V|)$ parameters and consideration of only direct, local connections (tendency of preserving only first order proximity).

4.2. Multihop distance preserving techniques

Cao et al. proposed GraRep which accepts as input the adjacency matrix A of the graph, maximum transition step (hops) K, log shifted factor β, and dimension of representation vector d. For each transition step, the k-step log probabilistic matrix is computed $A^k_{i,j}$ as shown in Eq. 14. This matrix is used as an approximation for the k-hop similarity matrix.

$$A^k_{i,j} = max(log(\frac{(A_{i,j}/d_i)}{\sum_{l \in V}(A_{l,j}/d_l)^k})^k - \alpha, 0). \tag{5}$$

The SVD of this matrix gives W^k which has representation of current vertices as column vectors. Concatenation of W^k for all k's gives the final graph representations [23]. Tang et al. proposed "LINE" which is suitable for multiple types of information networks: undirected, directed, and/or weighted. It preserves first-order proximity ($k = 1$) i.e. closeness by distance using tie strength as well as second-order proximity ($k = 2$) i.e. global level proximity using shared neighborhood. The authors argue that the objective function preserves both the local and global network structures. An edge-sampling algorithm is proposed instead of classical stochastic gradient descent to achieve optimization [9].

A second category of multihop based techniques use k-hop neighborhood overlap as a measure of similarity. Measures such as Adamic-Adar score or Jaccard similarity are used to construct a neighborhood similarity matrix S.

$$L = \sum_{(u,v) \in (V*V)} ||z_u^T z_v - S_{u,v}||^2 \tag{6}$$

Ou et al. proposed high-order proximity preserved embedding (HOPE) for preserving asymmetric transitivity in directed graphs is one such technique. It uses a neighborhood similarity matrix S calculated by Jaccard similarity and trains embeddings that capture this similarity in low dimension space. The objective function of HOPE is given in Eq. 7. It is the L_2-norm of the loss function which needs to be minimized. HOPE utilizes SVD to minimize the loss function [15].

$$min||S - U^s.{U^t}^T||_F^2, \tag{7}$$

$$S = M_g^{-1}.M_t, \tag{8}$$

$$M_g = I - \beta.A, \tag{9}$$

$$M_t = \beta.A, \tag{10}$$

where

1. S = high order proximity matrix in Eqs. 7 and 8
2. U^s, U^t = source and target embedding vectors
3. M_g, M_t = polynomial of adjacency matrix A in Eqs. 9 and 10
4. I = identity matrix, β = decay parameter of Katz index

Shi et al. demonstrated the applicability of neighborhood overlap matrix to calculate embeddings for heterogeneous information networks in 'ASPEM'. The authors argued that it mitigated the incompatibility among aspects via considering each aspect separately. Each aspect was defined as a unit representing one underlying semantic facet of the system. The applicability of this network was on graphs $G = (V, E)$ with a node type mapping $\phi : V \to T$ and an edge type mapping $\psi : E \to R$ where nodes are of multiple types $|T| > 1$ and edges $|R| > 1$ to are of multiple types [35]. Heterogeneous preference embedding model was proposed by Chen et al. for embedding a user-entity social network. The entities were song tracks, albums, singers which the user listened to. The edge weights of the social network represented the frequency of user to entity interactions i.e. rating or listening times as shown in Eq. 11. Neighborhood overlap between two users reflected their preferences towards the entities and the learned representations would reflect this [12].

$$Pr(v_j|\phi(v_i)) = \begin{cases} 1 & if v_j \in Context(v_i) \\ 0 & otherwise \end{cases} \tag{11}$$

$$O = -\sum_{i,j \in S} w_{i,j} logp(v_j|\phi(v_i)) + \lambda \sum_i ||\phi(v_i)||^2, \tag{12}$$

where

- $\phi(v_i)$ = vector representation of v_i
- w indicates the weight of the edge
- S is a set of sampling pairs
- λ weight to prevent overfitting
- $v_j \in Context(v_i)$ (in)direct connection to v_i or sequences of nodes in neighborhood
- $Pr(v_j|\phi(v_i))$ posterior probability
- O objective function optimized using stochastic gradient descent as shown in Eq. 12

LANE [10] has a joint objective function based on learning embeddings by utilizing two similarity matrices (S_1, S_2). S_1 is the adjacency matrix and S_2 is attribute similarity matrix calculated pairwise using Jaccard similarity.

$$L = \sum_{(u,v) \in (V*V)} ||z_u^T z_v - S_{u,v}^1||^2 + ||z_u^T z_v - S_{u,v}^2||^2 \tag{13}$$

AANE [11] utilizes a similarity matrix based on attribute matching of the pair of nodes to learn embeddings.

$$L = \sum_{(u,v)\in(V*V)} ||z_u^T z_v - ATT_{u,v}||^2 \tag{14}$$

Similar to adjacency-based measures, methods of this category of techniques also have to iterate over V^2 pair of nodes. In addition to that, deterministic node similarity measures have to be hand-designed which is the key drawback of multihop-based techniques.

4.3. Random walk cooccurrence preserving techniques

These techniques perform random walks on the graph starting from different nodes in the network. A multiset consisting of nodes visited on random walks is created. A three-layered neural network with a single hidden layer (no activation function) is used for generating embeddings from the random walks. The ANN is trained using one-hot-encoded (OHE) node vectors as shown in Figure 4. Each training pair is x, y where x is the OHE for the starting node of the random walk and $y \in R^{|V|}$ is vector of nodes such that $y^i = 1$ if i^{th} node is present in the random walk. The final layer of the neural network is a softmax layer that outputs the probabilities of nodes cooccurring on random walks from the input node (Eq. 15). The weights of the ANN are optimized by back-propagation to minimize the loss specified in Eq. 15. After the optimization, the weights of the hidden layer are the node embeddings [37]. The techniques of this category differ in the strategies used to run random walks on the original graph.

$$L = \sum_{(u\in V)} \sum_{(v\in N_r(u))} \frac{exp(z_u^T z_v)}{\sum_{n\in V} exp(z_u^T z_v)}. \tag{15}$$

Perozzi et al. proposed WALKLETS for learning multiscale representations of vertices in a network. The authors argue that multiscale relationships can be obtained by subsampling short random walks on the vertices of a graph. By ‘skipping’ over steps in each random walk, WALKLETS generates a corpus of vertex pairs which are reachable via paths of a fixed length. This corpus is then used to learn a series of latent representations using the ANN, each of which captures successively higher-order relationships from the adjacency matrix [38]. DeepWalk first generates random walks on the social network. Each walk contains many vertices in a line. Then, each walk is treated as a sentence and applied to the ANN in Figure 4. However, a drawback of DeepWalk is that it lacks a clear objective function and also it is designed only for networks with binary edges [19]. As DeepWalk uses plain random walks, it cannot capture second-order proximity of nodes, a modified strategy for running random walks was proposed in Node2vec by Leskovec et al.. Random walks used in Node2Vec alternate between depth-first (DFS) and breadth-first search (BFS) on the graph in a random manner. The authors speculated that such a hybrid strategy captured both first-order (using DFS) and second-order (using BFS) proximity. Node2vec’s sampling strategy accepts four arguments: Number of random walks to be generated from each node in the graph n; number of nodes in each walk l; P return hyperparameter; q in-out hyper parameter, context window size and number of iterations. The algorithm for the random walk generation will go over each node in the graph and generate n random walks of length l. If a random walker transitions from node t to node v the probability to transition from v to any one of his neighbors is $edgeweight * \alpha$ (normalized), where α is dependent on the hyperparameters. Using the sampling strategy, node2vec generates

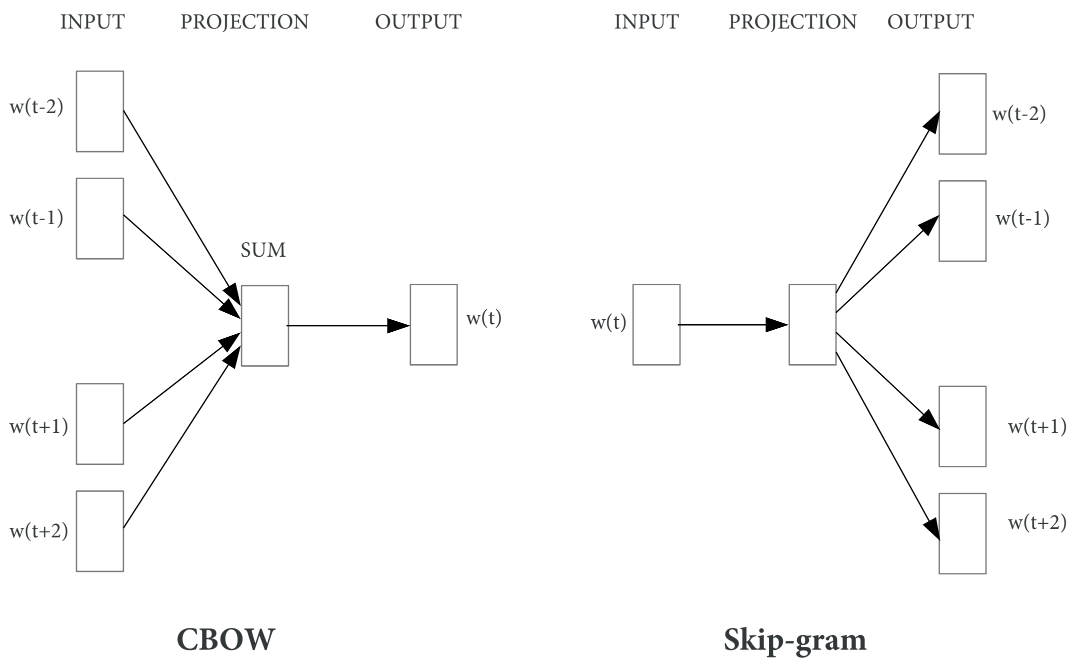

Figure 4. Use of ANN to generate node embeddings in random-walk-based models (word2vec or skipgram model). The CBOW architecture predicts the current word based on the context, and the Skipgram predicts surrounding words given the current word. The image was extracted from Mikolov et al. [37].

"sentences" (directed subgraphs) which are multisets of vertices. These are then used to train the word2vec model for obtaining embeddings [20].

Gat2vec proposed by Sheikh et al. decomposes the social network into two data structures viz. social network of connections G_s and bipartite graph of nodes and attributes G_a. It takes parameters ν_s, ν_a which are the number of random walks per node in G_s, G_a respectively, length of random walks on G_a, G_s given by λ_s, λ_a, context window c and embedding size d [22]. Random walks on both graphs are performed and the results are provided to Skipgram neural network to obtain the vector embeddings. Yang et al. proposed text-associated DeepWalk for derived network embedding preserving network structure and text information associated with nodes [39]. The authors prove the equivalence of DeepWalk and the Matrix factorization given in Eq. 16 (W, H are the learned embeddings):

$$\min_{W,H} ||M - W^T H||_F^2 + \frac{\lambda}{2}(||W||_F^2 + ||H||_F^2). \tag{16}$$

This equation is modified to include the text attribute information matrix T as given in Eq. 17. The vertex embeddings are formed by appending w with HT after the factorization in Eq. 17 is solved.

$$\min_{W,H} ||M - W^T HT||_F^2 + \frac{\lambda}{2}(||W||_F^2 + ||H||_F^2). \tag{17}$$

Random walk cooccurrence preserving techniques have $O(|E|)$ iterations and hence are considered to be more efficient than adjacency preserving and multihop distance preserving techniques. The other techniques belonging to these three categories of network representation learning techniques are listed in Table 1. The approaches shown in Table 1 require that all nodes in the graph should be present during training of the encoder. Hence, these approaches are inherently transductive in nature and do not naturally generalize to unseen nodes [40, 41]. The embedding matrix learned by these encoders is $Z = \mathbb{R}^{|V|*k}$. Each node of the social network is present in Z as a k-dimension column vector.

The reason behind calling these frameworks as "shallow encoder"-based frameworks is that to obtain the embedding of a node using the trained encoder, a simple look-up is needed. This is in contrast with "deep encoder"-based techniques that map the nodes to their embedding vectors using a complex nonlinear mapping function. As the parameters used by the "deep encoder"-based frameworks for mapping the nodes to their embedding vectors, they are more efficient compared to "shallow encoder"-based frameworks.

In summary, the "shallow encoder"-based NRL techniques fail to generate embeddings for nodes that were not present during training (absence of transductive nature). Moreover, none of such techniques consider unstructured data associated with the nodes in generating embeddings. Finally, there is an absence of parameter sharing that leads to additional parameters ($O(|V|)$) increasing computation time. To overcome these drawbacks, deep learning techniques that utilize graph convolutional neural networks, recurrant neural networks based on LSTM, and GRU were proposed.

Table 1. Shallow-encoder-based network representative learning frameworks.

Name	Optimization objective	Class of framework
Isomap [33]	Eq. 1	Adjacency
Locally linear embedding (LLE) [34]	Eq. 1	Adjacency
Landmark classical scaling [34]	Eq. 1	Adjacency
FSCNMF [13]	Eq. 1	Adjacency
LANE [10]	Eq. 1	Adjacency
AANE [11]	Eq. 1	Adjacency
HOPE [15]	Eq. 14	Multihop distance
VERSE [14]	Eq. 14	Multihop distance
HPE [42]	Eq. 14	Multihop distance
GraRep [23]	Eq. 3	Neighborhood overlap
LINE [9]	Eq. 3	Neighborhood overlap
ASPEM [35]	Eq. 4	Random walk cooccurrence
gat2vec [22]	Eq. 4	Random walk cooccurrence
DeepWalk [19]	Eq. 4	Random walk cooccurrence
Walklets [38]	Eq. 4	Random walk cooccurrence
struc2vec [43]	Eq. 4	Random walk cooccurrence
TADW [39]	Eq. 4	Random walk cooccurrence
graph2vec [44]	Eq. 4	Random walk cooccurrence
node2vec [20]	Eq. 4	Random walk cooccurrence

4.3.1. Deep-learning-based NRL models

Graph convolutional neural networks (GCNNs) are used to generate node embeddings based on aggregating information from local neighborhoods as given in Eq. 18. Every node in the social network has a unique computation graph. Neighborhood aggregation can be viewed as a center-surround filter [41, 45, 46] which is mathematically related to spectral graph convolutions [29, 30].

$$h_k^v = \sigma(W_k \sum_{u \in N(v)} \frac{h_u^{k-1}}{|N(v)|} + B_k h_v^{k-1}), \forall k > 0, \tag{18}$$

where

- h_v^k - k^{th} layer embedding of v
- σ - Nonlinear activation function
- $\sum_{u \in N(v)} \frac{h_u^{k-1}}{|N(v)|}$ - Average of neighbors previous layer embedding
- h_v^{k-1} - $k-1^{th}$ layer embedding of v
- W_k, B_k - Trainable matrices of weights and biases of layer k

The same aggregation parameters W_k, B_k are shared for all nodes. The number of model parameters is sublinear in $|V|$ and can be generalized to unseen nodes. The key distinction between GCN-based techniques is the method of aggregating neighbor information across the layers.

The GCN architecture proposed by Kipf et al. uses a variation of the neighborhood aggregation approach given by Eq. 19 [30]. Empirically, the authors found that the best results were obtained by using the same transformation matrix for self and neighbor embeddings. Compared to basic neighborhood aggregation, Kipf et al. used a normalization that varied across different neighbors.

$$h_k^v = \sigma(W_k \sum_{u \in N(v) \cup u} \frac{h_u^{k-1}}{\sqrt{|N(v)||N(u)|}}). \tag{19}$$

Hamilton et al. proposed GraphSAGE [40] that generates embedding for each node using a modified GCNN that relies on an aggregation function that maps two sets of vectors to a single vector as given in Eq. 20.

$$h_k^v = \sigma([W_k . AGG(h_u^{k-1}, \forall u \in N(v)), B_k h_v^{k-1}]). \tag{20}$$

Variants of GraphSAGE are mean, pool (transform neighbor vectors and apply symmetric vector function), and LSTM (apply LSTM to random permutation of neighbors). The key difference between these approaches is the type of aggregation function used in them as given in Eqs. 21–23.

$$AGG = \sum_{u \in N(v)} \frac{h_u^{k-1}}{|N(v)|}, \tag{21}$$

$$AGG = \gamma(Q h_u^{k-1}, \forall u \in N(v)), \tag{22}$$

$$AGG = LSTM([h_u^{k-1}, \forall u \in \pi(N(v))]). \tag{23}$$

4.3.2. ohmNet :

Zitnik et al. used graph convolutional neural network for constructing a link prediction system for medicine to side-effect prediction. The heterogeneous graph of drug-protein interaction was constructed with side-effects as the edges. The architecture has an encoder to generate graph embedding and a decoder to translate embedding information for predicting side-effects [31]. OhmNet [32] is an unsupervised feature learning technique for multilayer network.

$$P(v_i|v_j) = \frac{exp(u_i^T * u_j)}{\sum_{i' \in A} exp(u_{i'}^T * u_j)}. \tag{24}$$

The second-order proximity between a pair of vertices v_i, v_i' is given by $p(.|v_i), P(.|v_i')$. The conditional distribution $p(.|v_i)$ is to be matched to its empirical distribution $\hat{p}(.|v_i)$ which can be achieved by minimizing Eq. 25 using KL divergence $d(.,.)$. λ_j is the degree of the vertex and $\hat{p}(v_i|v_j) = \frac{w_{i,j}}{deg_j}$.

$$O = \sum_{j \in B} \lambda_j d(\hat{p}(.|v_i), p(.|v_i)), \tag{25}$$

$$O = -\sum_{i,j \in E} w_{i,j} logp(v_j|v_i). \tag{26}$$

Eq. 26 is optimized using stochastic gradient descent with negative sampling or edge sampling [47].

4.3.3. Semisupervised embedding in attributed networks with outliers (SEANO)

Liang et al. proposed a method to learn a low-dimensional vector representation that systematically captures the topological proximity, attribute affinity, and label similarity of vertices in a partially labeled attributed network. The architecture of the model consists of an artificial neural network with two input layers and two output layers. Nonlinear activation functions are used to transform the features into a nonlinear low-dimensional space. Back-propagation and mini-batch stochastic gradient are used to minimize the objective function.

4.3.4. VERtex similarity embeddings (VERSE)

Tsitsulin et al. proposed VERtex similarity embeddings (VERSE), a neural-network-based network embedding model. Given a graph, a similarity function sim_G and the embedding space dimensionality d, the output embedding matrix W is set to $N(0, 1/d)$. The objective function is optimized using gradient descent. This is done by repeatedly sampling a node from the positive distribution P, sample the sim_G (e.g., pick a neighboring node), and draw s negative examples. P and Q are $\sim U(1,n)$. However, the authors have not given the justification for the time $O(dsn)$ and space complexity $O(n^2)$; d is the latent space dimensionality, n the number of nodes, m the number of edges, and s the number of samples used. [14].

GCNN usually are 2-3 layers deep; hence, it is a challenge to build models with many layers of neighborhood aggregation [48]. The issues that are seen in GCNNs with multiple layers are overfitting and vanishing or exploding gradients during back-propagation. Li et al. used recurrent neural networks to resolve these issues. The authors proposed Gated GCNN approach that allowed for parameter sharing across various layers of the neural network. Gated GCNN used neighborhood aggregation with RNN state update as follows:

- Get "message" from neighbors at step k - $m_v^k = W \sum_{u \in N(v)} h_u^{k-1}$
- Update node "state" using gated recurrent unit (GRU) such that new node state depends on the old state and the message from neighbors - $h_v^k = GRU(h_v^{k-1}, m_v^k)$

This allows complex information about global graph structure to be propagated to all nodes.

5. Obstacles in deep learning architectures

Theoretically, deep neural networks are accurate and this accuracy increases with additional hidden layers [29]. However, practically in deep neural networks the problem of vanishing or exploding gradients is observed. This hampers convergence [49]. As the depth of the networks increases, the number of parameters also increase ($\sim 10^6 - 10^7$) and the accuracy during optimization gets saturated. Adding additional layers in such scenario hampers the training accuracy while increasing the training time.

5.1. Summary

Representative learning techniques have to reliably overcome challenges such as link sparsity, scalability, sparsity, and unreliability of side information, and have versatility to learn from complex graph structures.

The classical structure preserving techniques viz. SVD, PCA, NNMF tend to overfit the adjacency matrix and could not be used for applications that require capability to make inferences from network structure viz. link prediction. After Skipgram model was proposed in 2000, random-walk-based methods were developed. These were first-order as well as second-order proximity preserving. The chief drawback of these techniques was the presence of large number of parameters that made them parameter-dependent. GCNN-based techniques were proposed in 2016 and were found to match the results of state-of-the-art network embedding techniques. The key advantage of these methods was the general architecture of CNNs that was applicable across social networks of diverse domains. GCNN could be used with both social networks with and without attributes. Graph neural networks (GCNs), however, were found to be computationally expensive or requiring the use of heuristics to apply them to large graphs. It was also observed that GCN techniques like GraphSAGE that were proposed for massive graphs ($n > 10^6$) were not applicable on smaller or denser graphs. Another drawback of GCN (or ANN)-based techniques is that when the data is in the form of a graph structure, stochastic gradient descent for parameter tuning and back-propagation cannot be applied in a straightforward manner.

Network embedding methods have mushroomed in the literature; however, there is no universal acceptability to any particular technique as each has its own advantages and drawbacks. As longitudinal data becomes available on platforms such as Facebook, Twitter, and LinkedIn, dynamic network embedding frameworks too shall become a reality. Currently, however, most of the frameworks that analyze social networks are static (cross-sectional).

6. Datasets

6.1. Konect

Institute of Web Science and Technologies at the University of Koblenz–Landau provides a repository for network data from diverse sources such as social networking sites, gene expression, botany, e-commerce, affiliation, authorship, citation, folksonomy, human contact, communication, infrastructure, interaction and semantics [50, 51].

6.2. SNAP

SNAP is the network analysis platform in Python and C++ of Stanford University. It also has close to 50 large network datasets from diverse fields such as social networks, web graphs, transportation networks, hyperlink networks, citation networks, collaboration networks, and transmission networks [20].

7. Conclusion

Social networks have become ubiquitous as representation models for various real and synthetic systems. However, when data is represented in the form of a graph there are inherent challenges associated with processing it. These challenges are interdependency between data points and high dimensionality. To overcome these challenges, network science literature uses representation learning frameworks. With the use of such frameworks, the high-dimensional network structure is converted to 2D vector representations. In this form, the interdependency between data points no longer exists and downstream-network-based applications can be developed using machine learning.

With the abundance of network embedding frameworks available in the literature, it is necessary to understand the advantages and disadvantages or intuitions behind these methods before they can be applied for a particular case study. Proximity preserving methods require designing similarity measures and random walk-based methods require designing a strategy for running the random walker. Hence, these are parameter-dependent. Deep learning (DL)-based methods of NRL using convolutional neural networks, recurrent neural networks (LSTM and GRU) do not face such drawbacks. It is found that these frameworks achieve state-of-the-art representation learning accuracy on social networks and at the same time require less parameters due to parameter sharing. If challenges related to high computational complexity and optimization process are overcome, then DL-based methods can become the backbone of social network applications.

References

[1] Denny M. Social Network Analysis. Institute for Social Science Research, University of Massachusetts, Amherst, MA, USA: Academic Press, 2014.

[2] Pei J, Zhu W, Cui P, Wang X. A survey on network embedding. IEEE Transactions on Knowledge and Data Engineering 2018; 21: 126-139.

[3] Denny M. Intermediate Social Network Theory. Institute for Social Science Research, University of Massachusetts, Amherst, MA, USA: Academic Press, 2015.

[4] Hoff PD, Raftery AE, Handcock MS. Latent space approaches to social network analysis. Journal of the American Statistical Association 2002; 64: 1090-1098.

[5] Snijders TAB. Longitudinal methods of network analysis. Encyclopedia of Complexity and System Science 2009; 24: 5998–6013.

[6] Ferrara E, Galstyan A, Goyal P, Hosseinmardi H. Capturing edge attributes via network embedding. arXiv preprint 2018; arXiv:1805.03280.

[7] Hu X, Huang X, Li J. Label informed attributed network embedding. In: ACM 2017 Proceedings of the Tenth International Conference on Web Search and Data Mining; New York, NY, USA; 2017. pp. 731-739.

[8] Hu X Huang X, Li J. Accelerated attributed network embedding. In: SIAM 2017 Proceedings of the International Conference on Data Mining; Houston, TX, USA; 2017. pp. 633-641.

[9] Tang J, Qu M, Wang M, Zhang M, Yan J et al. Line: Large-scale information network embedding. In: WWW 2015 Proceedings of the 24th International Conference on World Wide Web; Florence, Italy; 2015. pp. 1067-1077.

[10] Huang X, Li J, Hu X. Label informed attributed network embedding. In: ACM 2017 Proceedings of the Tenth International Conference on Web Search and Data Mining; Cambridge, United Kingdom; 2017. pp. 731–739.

[11] Huang X, Li J, Hu X. Accelerated attributed network embedding. In: ACM 2017 Proceedings of the SIAM International Conference on Data Mining; Notre Dame, IN, USA; 2017. pp. 633-641.

[12] Zhang H, Chua TS, Liao L, He X. Attributed social network embedding. arXiv preprint 2017; arXiv:1705.04969.

[13] Bandyopadhyay S, Kara H, Kannan A, Murty MN. Fscnmf: Fusing structure and content via nonnegative matrix factorization for embedding information networks. arXiv preprint 2018; arXiv:1804.05313.

[14] Tsitsulin A, Mottin D, Karras P, Muller E. Verse: Versatile graph embeddings from similarity measures. In: WWW 2018 Proceedings of the Conference on World Wide Web; Lyon, France; 2018. pp. 539–548.

[15] Ou M, Cui P, Pei J, Zhang Z, Zhu W. Asymmetric transitivity preserving graph embedding. In: ACM 2016 Proceedings of the 22nd SIGKDD International Conference on Knowledge Discovery and Data Mining; San Francisco, CA, USA; 2016. pp. 1105–1114.

[16] Rozemberczki B, Davies R, Sarkar R, Sutton C. Gemsec: Graph embedding with self clustering. arXiv preprint 2018; arXiv:1802.03997.

[17] Rozemberczki B, Sarkar R. Fast sequence based embedding with diffusion graphs. In: Springer 2018 International Conference on Complex Networks; Cambridge, United Kingdom; 2018. pp. 99-107.

[18] Yang Z, Cohen WW, Salakhutdinov R. Revisiting semi-supervised learning with graph embeddings. arXiv preprint 2016; arXiv:1603.08861.

[19] Perozzi B, Al-Rfou R, Skiena S. Deepwalk: Online learning of social representations. In: ACM 2014 Proceedings of the 20th SIGKDD International Conference on Knowledge Discovery and Data Mining; Washington, DC, USA; 2014. pp. 701-710.

[20] Grover A, Leskovec J. node2vec: Scalable feature learning for networks. In: ACM 2016 Proceedings of the 22nd SIGKDD International Conference on Knowledge Discovery and Data Mining; San Francisco, CA, USA; 2016. pp. 855-864.

[21] Sheikh N, Kefato Z, Montresor A. gat2vec: representation learning for attributed graphs. Computing 2018; 9: 1-23.

[22] Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J. Distributed representations of words and phrases and their compositionality. Advances in Neural Information Processing Systems 2013; 23: 3111–3119.

[23] Cao S, Lu W, Xu Q. Grarep: Learning graph representations with global structural information. In: ACM 2015 Proceedings of the 24th International on Conference on Information and Knowledge Management; Melbourne, VIC, Australia; 2015. pp. 891-900.

[24] Liu Q, Li Z, Lui J, Cheng J. Powerwalk: Scalable personalized pagerank via random walks with vertex centric decomposition. In: ACM 2016 Proceedings of the 25th International on Conference on Information and Knowledge Management; Indianapolis, Indiana, USA; 2016. pp. 195-204.

[25] Pandhre S, Mittal H, Gupta M, Balasubramanian VN. Stwalk: learning trajectory representations in temporal graphs. In: ACM 2018 Proceedings of the Joint International Conference on Data Science and Management of Data; Goa, India; 2018. pp. 210-219.

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

[27] Lin Y, Liu Z, Sun M, Liu Y, Zhu X. Learning entity and relation embeddings for knowledge graph completion. In: AAAI 2015 Proceedings of the Twenty-Ninth Conference on Artificial Intelligence; Austin, Texas, USA; 2015. pp. 2181-2187.

[28] Wang Z, Ye X, Wang C, Wu Y, Wang C et al. Rsdne: Exploring relaxed similarity and dissimilarity from completely-imbalanced labels for network embedding. Network 2018; 11: 14-26.

[29] Zhang M, Cui Z, Neumann M, Chen Y. An end-to-end deep learning architecture for graph classification. In: AAAI 2018 Proceedings of Conference on Artificial Intelligence; New Orleans, LA, USA; 2018. pp. 531-538.

[30] Welling M, Kipf TN. Semisupervised classification with graph convolutional networks. arXiv preprint 2016; arXiv:1609.02907.

[31] Leskovec J, Zitnik M, Agrawal M. Modeling polypharmacy side effects with graph convolutional networks. arXiv preprint 2018; arXiv:1802.00543.

[32] Zitnik M, Leskovec J. Predicting multicellular function through multilayer tissue networks. Bioinformatics 2017, 33: 190-198.

[33] Balasubramanian M, Schwartz EL. The isomap algorithm and topological stability. Science 2002; 295: 7.

[34] Roweis ST, Saul LK. Nonlinear dimensionality reduction by locally linear embedding. Science 2000; 290: 2323-2326.

[35] Zhu Q, Kaplan L, Han J, Shi Y, Gui H. Aspem: Embedding learning by aspects in heterogeneous information networks. In: SIAM 2018 Proceedings of the International Conference on Data Mining; San Diego, California, USA; 2018. pp. 144-152.

[36] Liao L, He X, Zhang H, Chua TS. Attributed social network embedding. arXiv preprint 2017; arXiv:1705.04969.

[37] Ozer Z, Ozer I, Findik O. Diacritic restoration of Turkish tweets with word2vec. Engineering Science and Technology 2018; 21: 1120-1127.

[38] Perozzi B, Kulkarni V, Chen H, Skiena S. Don't walk, skip!: online learning of multiscale network embeddings. In: ACM 2017 Proceedings of the International Conference on Advances in Social Networks Analysis and Mining; Sydney, Australia; 2017. pp. 258-265.

[39] Zhao D, Sun M, Chang EY, Yang C, Liu Z. Network representation learning with rich text information. In: Twenty-Fourth International Joint Conference on Artificial Intelligence; Buenos Aires, Argentina; 2015. pp. 2111-2117.

[40] Hamilton W, Ying Z, Leskovec J. Inductive representation learning on large graphs. Advances in Neural Information Processing Systems 2017; 31: 1024-1034.

[41] Scarselli F, Gori M, Tsoi AC, Hagenbuchner, M, Monfardini, G. The graph neural network model. IEEE Transactions on Neural Networks 2005; 20: 61-80.

[42] Chen CM, Tsai MF, Lin YC, Yang YH. Query-based music recommendations via preference embedding. In: ACM 2016 Proceedings of the 10th Conference on Recommender Systems; Boston, MA, USA; 2016. pp. 79-82.

[43] Song L, Dai H, Dai B. Discriminative embeddings of latent variable models for structured data. In: ACM 2016 Proceedings of the 33rd International Conference on Machine Learning; New York, NY, USA; 2016. pp. 2702-2711.

[44] Narayanan A, Chandramohan M, Chen L, Liu Y, Saminathan S. Subgraph2vec: Learning distributed representations of rooted subgraphs from large graphs. arXiv preprint 2016; arXiv:1606.08928.

[45] Hamilton W, Ying Z, Leskovec J. Representation learning on graphs: methods and applications. arXiv preprint 2017; arXiv:1709.05584.

[46] Bronstein MM, Bruna J, LeCun Y, Szlam A, Vandergheynst P. Geometric deep learning: going beyond euclidean data. IEEE Signal Processing Magazine 2017; 34: 18-42.

[47] Mei Q, Tang J, Qu M. Pte: Predictive text embedding through large-scale heterogeneous text networks. In: ACM 2015 Proceedings of the 21th SIGKDD International Conference on Knowledge Discovery and Data Mining; Sydney, NSW, Australia; 2015. pp. 1165-1174.

[48] Li Y, Tarlow D, Brockschmidt M, Zemel R. Gated graph sequence neural networks. arXiv preprint 2016; arXiv:1511.05493.

[49] Chen J, Ma T, Xiao C. Fastgcn: fast learning with graph convolutional networks via importance sampling. arXiv preprint 2018; arXiv:1801.10247.

[50] Kunegis J. Handbook of network analysis [konect–the koblenz network collection]. arXiv preprint 2014; arXiv:1402.5500.

[51] Preusse J, Kunegis J. Fairness on the web: Alternatives to the power law. In: ACM 2012 Proceedings of the 4th Annual Web Science Conference; Evanston, IL, USA; 2012. pp. 175-184.