Dynamic Network Embedding Survey

Guotong Xue[a], Ming Zhong[a], Jianxin Li[b], Jia Chen[a], Chengshuai Zhai[a], Ruochen Kong[c]

[a]*School of Computer Science, Wuhan University, Wuhan, China*
[b]*School of Information Technology, Deakin University, Melbourne, Australia*
[c]*School of Computer Science, Rensselaer Polytechnic Institute, New York, USA*

Abstract

Since many real world networks are evolving over time, such as social networks and user-item networks, there are increasing research efforts on dynamic network embedding in recent years. They learn node representations from a sequence of evolving graphs but not only the latest network, for preserving both structural and temporal information from the dynamic networks. Due to the lack of comprehensive investigation of them, we give a survey of dynamic network embedding in this paper. Our survey inspects the data model, representation learning technique, evaluation and application of current related works and derives common patterns from them. Specifically, we present two basic data models, namely, discrete model and continuous model for dynamic networks. Correspondingly, we summarize two major categories of dynamic network embedding techniques, namely, structural-first and temporal-first that are adopted by most related works. Then we build a taxonomy that refines the category hierarchy by typical learning models. The popular experimental data sets and applications are also summarized. Lastly, we have a discussion of several distinct research topics in dynamic network embedding.

Keywords: dynamic network embedding, survey, data model, representation learning, taxonomy

1. Introduction

In the last few years, the static network embedding problem has been intensively studied, and a variety of techniques have been proposed to learn network topology and attributes[1, 2, 3, 4, 5, 6, 7]. However, the networks are dynamic, namely, evolving over time in the real world. Not only the node and edge attributes may change as time goes by, but more importantly, the topology would also change, like creating or removing nodes and edges. Let us consider the following real world dynamic networks as examples.

Example 1 (User-item network). *The user-item network illustrated in Figure 1 is usually used for recommendation in e-commerce. It is recently noticed that a user would interact with different kinds of item in different periods, which formulates different temporary patterns in the network. For instance, a bunch of bursty edges may be created due to the sales on special shopping events. Therefore, the recommendation engine should recognize such pattern in a particular period.*

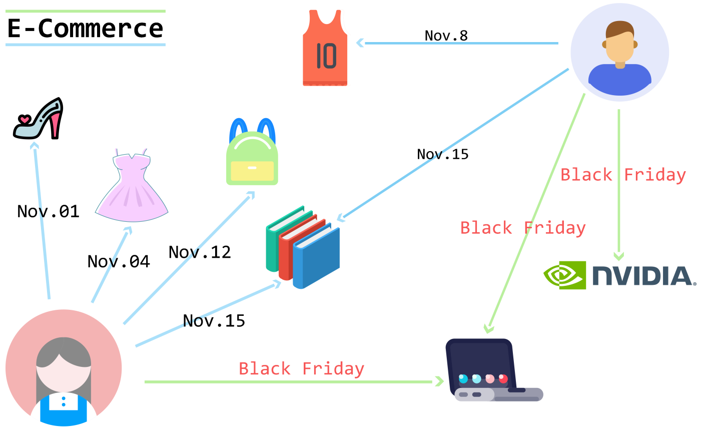


Figure 1: An illustration of user-item graph.

Example 2 (Payment network). *The pay-*

ment network is built from the electronic payment transactions in banks, credit card companies, and 3rd party payment institutions. Obviously, the payment network has constantly incoming nodes and edges every second. It has been leveraged to identify abnormal activities like financial fraud and money laundry in a real time fashion.

Example 3 (Social Network). *The social network is another highly dynamic network that evolves with online activities on social platforms. Normally, a user may have changing social interests, and thus may fall into different so-called short-term communities in the network. In order to better understand the evolving nature of human interactions, the social network analytics like link prediction and community detection should take the temporal information into consideration.*

Motivated by the essential dynamics of real-world networks, learning node representation for dynamic networks has become a more and more popular research topic recently. Fundamentally, the learned dynamic network embeddings should capture the network structure and reflect the temporal evolution. Namely, the learned embeddings should not only maintain the structural relationships between nodes in vector space, but also is required to describe the topological changes. However, it's hard to learn embeddings on dynamic network. We summarize the challenges as follows: Firstly, since the edges and nodes are evolving over time, temporal constraints are imposed on neighborhood aggregation methods. Besides, in order to model the dynamic of network, the learned node embeddings should also be functions of time to encode the temporal information. For example, considering that the time of node interactions are different, the impacts are also different. Furthermore, efficient learning methods are important since the scale of network may grow over time.

Though facing challenges above, the dynamic network embedding has the following significance compared with the static network embedding.

- **Capture temporal information** The temporal information is helpful for accurately analyzing network properties [8, 9]. However, static network embedding methods are not capable of dealing with it. Besides, embedding vectors may be reformed into different spaces if we learn them by rerunning static model several times. With learned dynamic network embeddings, researchers can utilize the additional information and better understand the network evolving process.

- **Update representation in a fine-grained granularity of time.** Instead of accumulating the historical changes of network and integrating them into a single static network, dynamic network embedding methods represent dynamic network as a series of snapshots or a sequence of new node/edge with timestamp. In this way, the node representation can be updated at the required granularity of time. What's more, with the support of RNN series models, the embeddings could be updated whenever a new interaction occurs. This could be applied in the scenarios requiring real-time performance.

- **Achieve high efficiency of updating representation.** The straightforward way of refreshing representation for dynamic networks is simply applying a static network embedding approach from scratch at each time step. However, retraining a network embedding model could be very time-consuming and cost high resources, especially for large scale networks. In contrast, most dynamic network embedding models could avoid retraining from scratch and achieve considerable update efficiency.

In this paper, we survey the state-of-the-art dynamic network embedding approaches. To the best knowledge we have, there is one published survey[10] and a preprint on arxiv.org[11] in this field. In contrast to them, our survey aims to 1) inspect the related works from more perspectives like data models, 2) build the taxonomy of existing techniques from higher abstract level based on underlying data models, and 3) have more discussion of common and principal issues involved in the related work like out-of-sample node embedding and prediction of future embedding. Compared with [10], our survey hopes to serve the researchers who have good knowledges of network embedding and just wish to know the recent progress on dynamic networks with more concise description and more practical taxonomy. Therefore, besides the summarization of related work, our survey would organize and compare them in several important aspects, and derive general and significant patterns from them, in order to assist the future work on dynamic network embedding.

Our contributions are summarized as follows.

- We present two common network data models underlying existing dynamic network embedding approaches, and propose a taxonomy that organizes these approaches by high-level methodologies and then by concrete learning models. The two methodologies summarized by us are actually two natural ways of handling the two data models respectively.
- We conduct comprehensive review of recent progress in dynamic network embedding, and make necessary comparison between them.
- We discuss several distinct and critical challenges in dynamic network embedding and explain how relevant existing approaches address them.
- We summarize the useful resources in existing approaches, like experimental datasets, evaluation tasks and real applications of dynamic network embedding.

The rest of this article is organized as follows. Section 2 presents the common data models used by related works. Section 3 gives a taxonomy of embedding techniques used by related work. Section 4 conducts discussion of challenging issues. Section 5 introduces commonly used datasets, evaluation, and practical applications.

2. Dynamic Network Models

In this section, we will introduce the data models of dynamic networks. Unlike the static network embedding approaches that almost follow a uniform network data model, the dynamic network embedding approaches have quite different definitions of dynamic network, which have significant impact on the selection of embedding techniques (as discussed in Section 3). Therefore, it is necessary to investigate the dynamic network models.

Basically, a static network is defined as a (attributed) network.

Definition 1 (Network). *A (attributed) network is represented as $G = (V; E; A)$, where V is a set of vertices or nodes, $E \subseteq V \times V$ is a set of edges between the nodes, and optionally, for each node $v \in V$, $A(v) \in \mathbb{R}^n$ is an attribute vector.*

The dynamic networks are graphs that have nodes, edges and attributes updated gradually over time. Naturally, there are two ways to update graphs, namely, discrete updates time interval by time interval and continuous updates in a stream. Correspondingly, there are two typical variants of the network data model that represent the dynamic networks. Next, we formally present these two models respectively.

2.1. Discrete Model

Most related works[12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26] use the discrete model to represent the dynamic networks. Intuitively, they see a dynamic network as a sequence of network snapshots, namely, independent and complete graphs. Normally, as figure 2 showed, the structure of graphs changes a bit between adjacent snapshots, which means a few of nodes and edges will be created or removed between snapshots.

We formally define the discrete model as follows.

Definition 2 (Discrete Model). *A discrete model of dynamic network G is a sequence of network snapshots within a given time interval.*

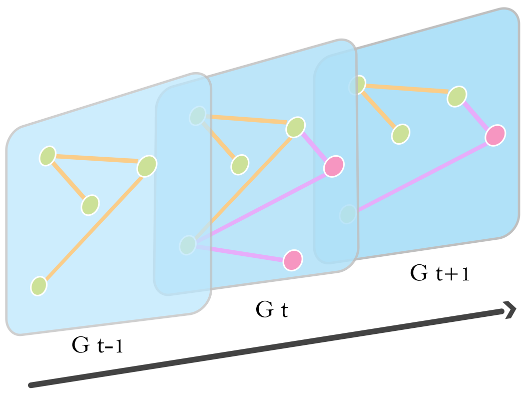
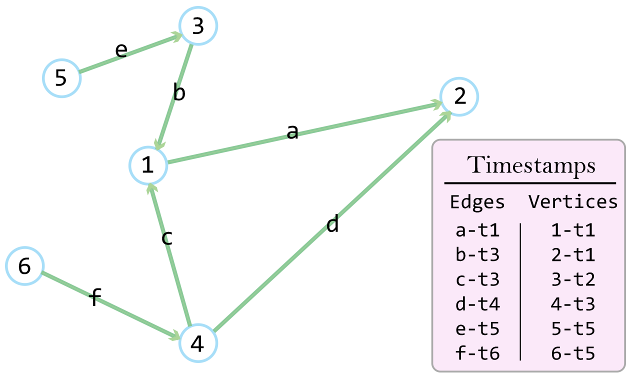


Figure 2: An illustration of discrete model.

We have $G = \{G_1, \ldots, G_T\}$, where T is the number of snapshots. Each snapshot $G_t = (V_t, E_t)$ is a static network recorded at time t.

2.2. Continuous Model

In contrast, other related works[27, 28, 29, 30, 31, 32, 33] use the continuous model, in which new nodes and edges are added in a stream manner. New edges are created and annotated with timestamps when interactions between nodes occur, which is illustrated in figure 3. Also, all nodes update their own timestamps when they are created or their properties changed.



Timestamps

Edges	Vertices
a-t1	1-t1
b-t3	2-t1
c-t3	3-t2
d-t4	4-t3
e-t5	5-t5
f-t6	6-t5

Figure 3: An illustration of continuous model.

We formally defined the continuous model as follows.

Definition 3 (Continuous Model). *A continuous model of dynamic network G is a network with edges and ndoes annotated with timestamps. We have $G = (V_T, E_T, \mathcal{T})$ where $\mathcal{T} : V, E \to \mathbb{R}^+$ is a function that maps each edge and node to a corresponding timestamp.*

In addition, it should be noted that the exactly used network model could be slightly different between each approach. For example, some methods consider network with attributes[15, 28, 18, 12, 21, 29, 31, 32, 23, 34, 24, 33, 26] while others[19, 16, 27, 30, 22, 20, 25] only deal with a raw network. What's more, as we summarized in table 1, there are some methods that can only tackle with adding nodes in the evolution while leave the deletion of nodes in future development.

3. Taxonomy of Embedding Techniques

In this section, we present a taxonomy of dynamic network embedding techniques. To explain the design of our taxonomy, it is needed to clarify the following fundamental challenges of learning dynamic network embedding firstly.

- **Preserve the networks' structure (and properties).** For the structure of the network, the learned embeddings should reflect the neighbor relationship between nodes and preserve high-order proximity. For example, nodes in same community should have similar embeddings .
- **Preserve the evolution of networks in long-term.** For the evolution of network, the learned embeddings should reflect the interaction with other nodes over time.

To address both the above challenges in one shot, there are naturally two highly abstract methodologies. Firstly, import the temporal features of dynamic network into the learning models that originally preserve the structure (and property) information of static networks, such as skip-gram[27, 31, 20, 16], auto-encoder[17, 30, 24, 25], GNN[15, 21, 26, 23, 34], etc. We call this methodology as structural-first. Secondly, import the structure (and property) features of static networks into the learning model that originally preserve the temporal information of dynamical data, such as RNN, LSTM[32, 28, 29, 19], etc. We call this methodology as temporal-first.

In fact, most related works adopt these two methodologies. Therefore, we categorize the dynamic network embedding techniques into three categories: structural-first, temporal-first, and others. In each category, there would be more specific sub-categories corresponding to different learning models. For example, Skip-gram based model, matrix factorization based model, auto-encoder based model and GNN based model are the sub-categories of structural-first. The complete taxonomy is shown in table 1.

3.1. Structural-first model

3.1.1. Matrix factorization based model

Matrix factorization techniques are one of the most traditional but effective ways to learn network embedding[38, 39]. We usually hope to find some lower dimensional matrices whose product serves as a approximation to the original matrix.

Matrix factorization based models often consider dynamic network as a constant changing matrices. Therefore, they could derive the embedding vectors through generalized SVD. Moreover, the matrix perturbation theory[40] is also adopted to update the embedding results.

[12] considers the raw representations of network topology and node attributes are different. Either of these could be incomplete and noisy. Therefore, they propose a dynamic attributed network embedding framework *DANE*. To get initial embedding of network $\boldsymbol{Y_A^{(t)}}$, they solve a generalized eigen-problem $\mathbf{L}_{\mathrm{A}}^{(t)}\mathrm{a} = \lambda \mathbf{D}_{\mathrm{A}}^{(t)}\mathrm{a}$, where a is the eigenvector and $\mathbf{Y}_{\mathrm{A}}^{(t)} = [\mathrm{a}_2, \ldots, \mathrm{a}_k, \mathrm{a}_{k+1}]$. The initial embedding of attributes $\boldsymbol{Y_X^{(t)}}$ is obtained in the same way. In order to resolve noisy data problem, they maximize the two embedding's correlation to get a consensus embedding $\boldsymbol{Y^t}$. Furthermore, to capture dynamic information, at snapshot t, the variation of the eigenvalue λ_i is derived as follows:

$$\Delta\lambda_i = \mathrm{a}_i'\Delta\mathrm{L}_{\mathrm{A}}\mathrm{a}_i - \lambda_i\mathrm{a}_i'\Delta\mathrm{D}_{\mathrm{A}}\mathrm{a}_i. \tag{1}$$

where $\boldsymbol{A^t}$ is the adjacency matrix, $\Delta\mathrm{D}_{\mathrm{A}}$ and $\Delta\mathrm{L}_{\mathrm{A}}$ are the perturbation of the diagonal matrix and Laplacian matrix.

And the perturbation of eigenvector a_i is given as follows:

$$\begin{aligned}\Delta\mathrm{a}_i = &-\frac{1}{2}\mathrm{a}_i'\Delta\mathrm{D}_{\mathrm{A}}\mathrm{a}_i\mathrm{a}_i \\ &+ \sum_{j=2, j\neq i}^{k+1}\left(\frac{\mathrm{a}_j'\Delta\mathrm{L}_{\mathrm{A}}\mathrm{a}_i - \lambda_i\mathrm{a}_j'\Delta\mathrm{D}_{\mathrm{A}}\mathrm{a}_i}{\lambda_i - \lambda_j}\right)\mathrm{a}_j\end{aligned}. \tag{2}$$

the i-th eigen-pair $(\Delta\lambda_i, \Delta\mathbf{x}_i)$ of node attributes can also be updated in similar manner.

Similarly, [14] also adopts the generalized SVD to learn representation of network. Specifically, they try to preserve high-order proximity by minimizing the following function:

$$\min \left\|\mathbf{S} - \mathbf{U}\mathbf{U}'^{\top}\right\|_F^2. \tag{3}$$

, where $\mathbf{U}, \mathbf{U}' \in \mathcal{R}^{N\times d}$. In matrix decomposition, $\mathbf{U}$ and $\mathbf{U}'$ can be seen as the basis and the coordinate. S is Katz Index, which is used to measure high order proximity. For incremental updating embedding results, they propose a generalized eigen perturbation to calculate the change of the eigenvalues and eigenvectors.

Though incremental matrix factorization methods could update the embedding results, [13] observes that such approximations[12] would lead to accumulated errors. Therefore, they design a approach to optimally set the restart time of model by deriving a lower bound of SVD minimum loss and setting a maximum tolerated error as threshold.

Moreover, [41] proposes *DyHNE* to learn node embeddings on heterogenous network. Specifically, they introduce meta-path based first- and second-order proximities to obtain heterogenous structural information. And following [14], *DyHNE* also updates learned node embeddings by employing eigenvalue perturbation.

In summary, matrix factorization based methods model dynamic network in discrete way and learn node embedding by solving generalized eigen-problem. Thanks to the matrix factorization mechanism, this type of model can flexibly handle the increase and decrease of nodes/edges. Besides, this type of model is usually based on matrix perturbation to capture dynamics, which means it is suitable for dynamic graphs with subtle changes. In addition, to reduce the computing

Method	Methodology	Data Mode			Learning Techniques				Experimental tasks		
		Model Type	Node/Edge Addition	Node/Edge Deletion	Matrix Factorization	Skip-gram	Deep Learning	Others	Node Classification	Link Prediction	Others
DANE[12]	S-first	discrete	both	both	yes				yes		network clustering
DHPE[14]	S-first	discrete	both	both	yes				yes	yes	high-order proximity approximation
Know-Evolve[28]	T-first	continuous	both				yes			yes	time prediction
STGCN[15]	S-first	discrete					yes				traffic prediction
DNE[16]	S-first	discrete	both	both		yes			yes		network layout
DynGEM[17]	S-first	discrete	both	both			yes			yes	network reconstruction
DepthLGP[18]	Others	discrete	both					yes	yes	yes	
DGNN[19]	T-first	discrete	both				yes		yes	yes	
dynnode2vec[20]	S-first	discrete	both	both		yes			yes	yes	anomaly detection
Continuous-Time Dynamic Networks[27]	S-first	continuous	edge			yes				yes	
DySAT[21]	S-first	discrete	edge	edge			yes			yes	
DyRep[29]	T-first	continuous	both				yes			yes	time prediction
Netwalk[30]	S-first	continuous	edge	edge			yes				anomaly detection
DynamicTriad[22]	Others	discrete	both					yes	yes	yes	link reconstruction
HTNE[31]	S-first	continuous	edge			yes			yes	yes	
JODIE[32]	T-first	continuous	edge				yes			yes	user state change prediction
EvolveGCN[23]	S-first	discrete	both				yes		yes	yes	edge classification
BurstGraph[24]	S-first	discrete	both				yes			yes	
AddGraph[34]	S-first	discrete	edge				yes				anomaly detection
dyngraph2vec[25]	S-first	discrete	edge	edge			yes			yes	
TGAT[33]	Others	discrete	both				yes		yes	yes	
DyHAN[26]	S-first	discrete	both	both			yes			yes	
DyHATR[35]	T-first	discrete	both	both			yes			yes	
THIGE[36]	T-first	continuous	edge	edge			yes				next-item recommendation
MMDNE[37]	Others	continuous	both				yes	yes	yes	yes	network reconstruction

T-first means temporal first while **S-first** means structural first.

Table 1: A Summary of Dynamic Network Embedding Methods

complexity during matrix perturbation process, accelerated solutions need to be developed.

3.1.2. Autoencoder based model

The idea of autoencoder was firstly introduced in [42]. Figure 4 provides a simply illustration of autoencoder mechanism. The encoder maps input samples to the feature space and generates latent representations while the decoder map them back to obtain the reconstructed samples.

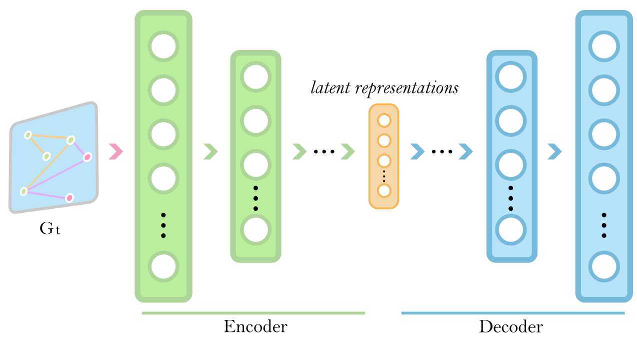


Figure 4: An illustration of autoencoder mechanism.

We could learn embeddings by minimizing the reconstruction error. Most of autoencoders are implemented through various types of neural network.

Representing the dynamic graphs as a collection of snapshots, *DynGEM* [17] employs a deep autoencoder as its core and incrementally learn the embedding of snapshot t from the embedding of previous snapshot $t-1$. Specifically, *DynGEM* initializes embedding at time t with the learned embedding from time $t-1$, then carries out model training. To work with growing number of nodes, they bring up a heuristic method to calculate the new sizes of neural network layers at each time step and add new layers if needed. The loss function is aimed to preserve structural proximity with following form:

$$L_{net} = L_{glob} + \alpha L_{loc} + \nu_1 L_1 + \nu_2 L_2. \qquad (4)$$

$L_{loc} = \sum_{i,j}^{n} a_{ij} \|\boldsymbol{y_i} - \boldsymbol{y_j}\|_2^2$ is the first-order proximity, where y is embedding value and A is adjacency matrix. And $L_{glob} = \sum_{i=1}^{n} \| (\hat{\boldsymbol{x}}_{\boldsymbol{i}} - \boldsymbol{x_i}) \odot \boldsymbol{b_i} \|_2^2 = \| (\hat{X} - X) \odot B \|_F^2$ is the second-order proximity which is preserved by an unsupervised reconstruction of the neighborhood of each node. b_i

is a vector with $b_{ij} = 1$ if $a_{ij} = 0$ else $b_{ij} = \beta > 1$.

Instead of using network as input, *NetWalk* [30] takes a lists of vertices sampled by random walk. Considering the sparsity of the input and output vectors, which are one-hot encoded, *Netwalk* applies a sparse autoencoder to learn the embedding. To be specific, KL-divergence is included as sparsity constraint in cost function, which measures the difference between two probability distributions. In order to learn the network structure, they also design a loss term to minimize the pairwise distance among all vertices of each random walk.

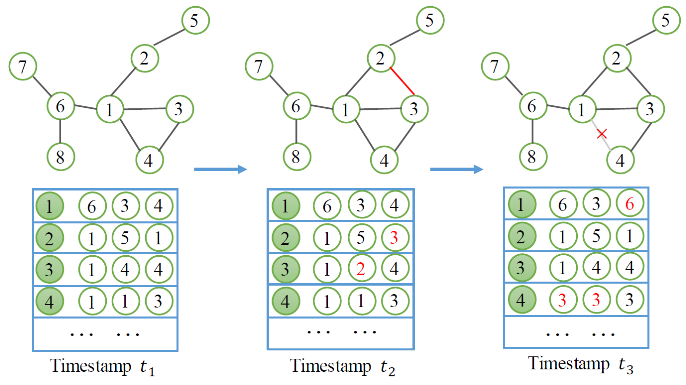


Figure 5: An illustration of updating the reservoirs. Figure extracted from [30]

Netwalk treats the edges addition/deletion in a continuous view, which means the changes come like a stream. Thus, *Netwalk* maintains a reservoir to save random walk results for each vertex, which is showed in figure 5. The content of reservoir would update with probability when edge changing occurs. Then they continue to train the model with the updated network walk set in a warm-start fashion.

[25] observes that *DynGEM*, [22] and *Timers* only utilize the previous time step network, which imply they assume the patterns are of short duration. Besides, *DynGEM* and *Timers* also assume that the changes are smooth and impose a regularizer to ensure temporal smoothness. Based on the above observations, *dyngraph2vec* hopes to use longer-term historical information to learn node embeddings. Therefore, *dyngraph2vec* takes l snapshots as input and predict the $t+l+1$ network. It integrates dense layer and LSTM layer as encoder to capture dynamic information. Moreover, [24] based on observation of user buying behavior, propose *BurstGraph* to especially capture bursty network changes. *BurstGraph* uses *GraphSage* [43] as encoder while utilizes RNN as a part of decoder to tackle with the evolution of network. Specifically, *BurstGraph* introduces a spike-and-slab distribution as an approximation of posterior distribution in the variational autoencoder framework[44] to deal with bursty links.

In short, the above methods show that the autoencoder as a general framework can learn the node representation on the dynamic network. The main difference between the methods is how to better capture temporal information. In particular, it is necessary to design different encoder according to different network change modes, such as smooth changes.

3.1.3. Skip-gram based model

Introduced in *Word2Vec*[45], Skip-gram is a simple but effective model to learn embedding representations. It's originally applied in word embedding task, which uses a input word to predict corresponding context. *DeepWalk* [4] generates sequences of vertices by random walk. Then, by treating walks as the equivalent of sentences, *DeepWalk* utilized Skip-gram model to learn node embeddings. *Node2Vec*[46], *LINE*[3] and other models make further progress to better capture network structure.

To generalize Skip-gram based network embedding techniques to dynamic field, [16] propose a decomposable objective equivalent to the objective of *LINE*. The adjusted objective function could learn latent representations for new vertices in snapshot. Besides, [16] also analyzes the influence of network evolution and design a selection mechanism to choose the greatly affected nodes to update.

Also tackling with a sequence of snapshots, [20] modified *Node2Vec* to learn dynamic network embedding. As repeatedly generating random walk for every snapshot is time-consuming, *dynnode2vec* only update the list of random walk for evolving nodes. The evolving nodes in the

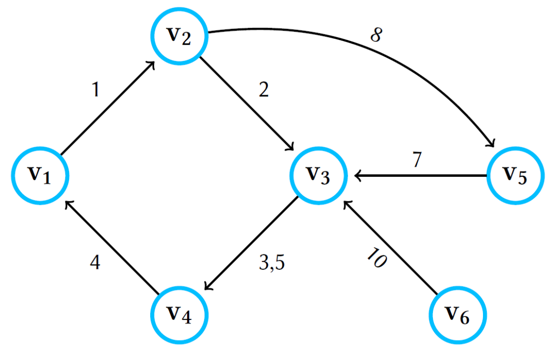


Figure 6: v_2,v_3,v_4 is a valid walk while v_5,v_3,v_4 isn't a valid walk since edge between v_5,v_3 is created later than edge between v_4,v_3. Figure extracted from [27]

Taxonomy

timestamp t are defined as:

$$\Delta V_t = V_{add} \cup \{v_i \in V_t \mid \exists e_i = (v_i, v_j) \in (E_{add} \cup E_{del})\}.$$

Similarly to *DynGEM*[17], *dynnode2vec* inherits parameters' weight from previous model(snapshot $t-1$), and train the current model with updated random walks.

However, [27] argues that representing dynamic network as a series of snapshots leads to coarse and noisy approximation. They suggest to model the temporal network as a continuous model. Such modeling approach may not only reduce the loss of information, but also let researchers could flexibly select appropriate granularity. That is to say researchers could customize the time interval. To be specific, as all the coming edges are annotated with timestamps, [27] add temporal constraint to random walk. The walk should traverse along edges in the direction of increasing time, which is illustrated at figure 6. In order to perform temporal random walks, the selection of initial edges and time has also been discussed. In addition, with the assumption that the nodes with new arriving edges are impacted by historical events, *HTNE*[31] applies Hawkes process to model the neighborhood formation.

In summary, Skip-gram based methods are often based on random walks to design sampling methods, thereby updating the context information of the node in network evolution. Benefiting from updating only a subset of nodes, this type of method exhibits higher performance when the network structure changes little.

3.1.4. GNN based model

Recently, applying neural network on network has made a booming development. Several kinds of graph neural network have been proposed and achieved impressive results on relevant tasks. Most of the models extend deep learning approaches for network data. For example, *GCNs* [47, 48, 49, 5, 43] realize convolution-like operation on graphs, attention mechanism is also applied in graph neural network[50, 51].

STGCN[15] focuses on solving the traffic prediction problem on traffic graphs. Due to all the positions of monitor station are fixed, a dynamic traffic network has a constant size of nodes. As time goes by, the node's features would keep changing, which reflect the traffic condition. *STGCN* adopts the first-order approximation of spectral graph convolutions[48] to gather the spatial information. Hoping to respond quickly to dynamic changes, instead of using RNN-based architecture, *STGCN* design a temporal convolution layer to capture dynamic behaviors of traffic flows. It integrates 1-D causal convolution and gated linear units.

However, network model in *STGCN* is a special case of dynamic network without growing size. [21] proposes *DySAT*, which is able to learn embedding from more general form of dynamic network. Observed that previous approaches[22, 12], which pursue smoothness of node representations, may ignore distinct temporal behaviors, [21] introduces self-attention mechanism. It has a clear architecture: model consists of a structural attention block and a temporal attention block. In structural block, *GAT*[50] is modified to collect spatial information for each snapshot and output a sequence of node embeddings at different time steps. Every node's embedding sequence is push forward to temporal attention block to learn the final representation. Benefiting by flexible attention weight, model could capture temporal dependencies at a fine-grained node-level granularity, rather than output smooth expressions for nodes. Besides, inspired by *DySAT*, *DyHAN*[26] uses hierarchical attention to learn node embedding, which performs better than *DySAT* experiment. It is worth mentioning that *DyHAN* also tackles with the structural heterogeneity of network. And experimental results show the efficacy of considering heterogeneity.

Since graph neural network could only gather neighborhood's features and provide embedding for static network, RNN-like techniques are also integrated to learn temporal information in some models. However, it's doubtful that whether RNNs could learn a valid embedding for new coming nodes in consideration of lacking historical features. In order to resolve this challenge while make use of RNNs to capture dynamic information at the same time, [23] proposes *EvolveGCN*. Basically, it uses GCN to generate node embeddings for every snapshot. Then, instead of directly applying recurrent layer to refine the embeddings, *EvolveGCN* uses RNN to train the weights of GCN. It provides GRU version and LSTM version of model, which could be written as:

$$\underbrace{\overbrace{W_t^{(l)}}^{GCN\ weights}}_{hidden\ state} = GRU(\underbrace{\overbrace{H_t^{(l)}}^{node\ embeddings}}_{input}, \underbrace{\overbrace{W_{t-1}^{(l)}}^{GCN\ weights}}_{hidden\ state}), \tag{5}$$

$$\underbrace{\overbrace{W_t^{(l)}}^{GCN\ weights}}_{output} = LSTM(\underbrace{\overbrace{W_{t-1}^{(l)}}^{GCN\ weights}}_{input}). \tag{6}$$

Though it is fancy to maintain temporal information through evolving parameters, *EvolveGCN*'s performance in link prediction is not outstanding.

Focusing on anomaly detection, [34] considers that *NetWalk*[30] fails to capture the long-term and short-term patterns. Therefore, [34] propose *AddGraph*, which also integrates RNN as part of model. At time t, current state representation is propagated by GCN,

$$\text{Current }^{t} = \text{GCN}_L\left(\mathbf{H}^{t-1}\right). \tag{7}$$

, where $\mathbf{H^{t-1}}$ is hidden state learned at time $t-1$. To collect more historical information, a contextual attention-based model[52] is applied. The model could be briefly written as a function:

$$\text{Short}^t = \text{CAB}\left(\mathbf{H}^{t-\omega}; \ldots; \mathbf{H}^{t-1}\right) \tag{8}$$

, where ω is the size of window. Then, *AddGraph* applies GRU to encode current state and historical resources as:

$$\mathbf{H}^t = \text{GRU}\left(\text{ Current }^{t}, \text{ Short }^{t}\right) \tag{9}$$

. As labeled data is especially insufficient in anomaly detection, negative sampling strategy and margin loss technique are also applied. What's more, other dynamic network embedding models[19, 24] also integrate GNN approaches to better extract structural features.

In general, unlike previous methods, the methods based on graph neural network can easily process attributed dynamic network. The difference is the strategy of capturing the network evolution. Some methods use parameter reuse to implicitly preserve temporal information, while others use attention mechanisms or RNNs to explicitly encode temporal information. In addition, because such methods are updated based on snapshots, it is not easy to perform more fine-grained and more timely node representation learning.

3.2. Temporal-first model

3.2.1. RNN based model

Based on [53], recurrent neural network(RNN) is very general form of neural network. RNN is designed for processing sequential data, where connections between neurons form a directed network along a temporal sequence. Outstanding models like GRU[54], LSTM[55], use neuron's internal state as memory to process variable length sequences of inputs.

As RNN's intrinsic ability to encode temporal information, simple RNN units are adopted to learn entity embedding over dynamic knowledge graph[28]. *DGNN*[19] try to learn the global impact of new edges while also consider local influence for new connected nodes. In essence, *DGNN* wants to update embedding of source node and endpoint when new interaction occur. Besides, the influence should also propagate to the first-order neighbors. Based on above motivations, it makes extension of LSTM, which can take embeddings of first-order neighbor and endpoint of a edge as input respectively.

Similarly, based on the observation of social network, [29] points out that the evolution of network contains two fundamental processes: association process and communication process. The former one represents the topology change which establishes or destroys a long-lasting bridge for information exchange between nodes, while the latter process means information propagates for a short time at adjacent nodes. To model the processes above, *DyRep* model dynamic network in continuous way and implement a RNN-like structure with attention mechanism. Instead of directly give embedding as output, the model learns functions to compute node representations, which is capable of inferring representation for unseen node.

[32] also adopts the continuous model. In particular, the model *JODIE* is customized for User-Item bipartite network, which is a common pattern in recommendation system. *JODIE* builds up a coupled RNN architecture and adds a attention layer to get final representation of user. It uses two separate RNN to update the embeddings of user and item for every interaction occurs. Take the update of user embedding as example, it could be formally written as:

$$u(t) = \sigma\left(W_1^u u\left(t^-\right) + W_2^u i\left(t^-\right) + W_3^u f + W_4^u \Delta_u\right). \tag{10}$$

, where $u(t^-)$, $i(t^-)$ are previous embedding of user and item, f is embedding of static features and Δ_u denotes the time since u's previous interaction. In this way, *JODIE* makes use of stationary properties and dynamic information in a unified framework. What's more, according to life experience, user's intent would not stay constant over time even no interaction with item occurred. This requires the updating embeddings could predict the trajectory of moving intent when there is no fresh information of nodes. In fact, the prediction of trajectory could be considered as embedding projection. And a linear operator is proposed to accomplish it. Therefore, the final embedding of user at time $t + \Delta$ is predicted as:

$$\begin{aligned}\tilde{j}(t+\Delta) =& W_1\widehat{u}(t+\Delta) + W_2\bar{u} \\ &+ W_3 i\left(t+\Delta^-\right) + W_4\bar{i} + B.\end{aligned} \tag{11}$$

, where $\widehat{u}(t+\Delta)$, $i\left(t+\Delta^-\right)$ are the projected user embedding and item embedding.

In addition, heterogeneous dynamic graphs have also received more and more attention. [35] proposes *DyHATR*, which uses the hierarchical attention model to capture the heterogeneity and introduces the temporal attentive GRU/LSTM to model the evolutionary patterns among snapshots. While [36] focus on next-item recommendation problem, they learn embedings on temporal heterogenous User-Item bipartite network. This kind of network has multiple types of edges, representing different interaction behaviors. Specifically, they divide the user's historical interactions into long-term interactions and short-term interactions based on timestamps. They introduce heterogeneous self-attention mechanism to learn long-term interaction while use GRU to learn users' current demands.

In short, RNN based methods usually adopt continuous model, which can update node and edge changes more timely. This type of method is usually used in recommendation systems. The main difference between each method is how to design a node representation learning method according to a specific network structure, such as a User-Item bipartite network.

3.3. Others

There are some works learn the dynamic network embedding in their unique ways. Facing the challenge of inferring embedding for out-of-sample nodes, *DepthLGP* is proposed by [18]. They design a high-order Laplacian Gaussian process (hLGP) to encode network properties and use a deep neural network to perform nonlinear transformation. Benefiting from nonparametric probabilistic modeling, *DepthLGP* maintains high speed for scalable inference.

Besides, *DynamicTriad*[22], a semi-supervised algorithm, learns embeddings by modeling the triadic closure process. Triadic closure is the property among three nodes, where an open triad is two of three nodes have connections and a closed triad means connection exists between any two nodes. They define the probability that the open triad (vi, vj, vk) evolves into a closed triad as

$$P_{\mathrm{tr}}^t(i,j,k) = \frac{1}{1+\exp\left(-\left\langle \boldsymbol{\theta}, \boldsymbol{x}_{ijk}^t \right\rangle\right)}, \tag{12}$$

where x_{ijk}^t is vector calculated using embeddings of node i, j, k, and θ is a social strategy parameter. In order to learn node embeddings, corresponding loss function and sampling strategy are proposed.

Moreover, using continuous data model, [37] studies network dynamics from the micro and macro perspectives and proposes *MMDNE*. Micro-dynamics describes the specific process of network structure changes, such as newly established edges, while macro-dynamics describes the evolution pattern of network scale. Specifically, they introduce temporal point process to describe the establishment of edges, i.e. micro-dynamics, and combine the hierarchical temporal attention to learn node representations. And for macro-dynamics, they define a general dynamics equation parameterized with network embeddings

In addition, in order to generate representations in an inductive fashion, temporal graph attention(*TGAT*)[33] proposes a functional time encoding technique by applying Bochner's theorem. Self-attention mechanism is also been applied as building block to aggregate neighborhood information.

4. Discussion

In this section, we discuss several critical challenges in learning dynamic network embeddings and correspondingly proposed methods.

- **Long-term Features Preservation**

  In the process of network evolution, special patterns such as communities and interactive features may be formed, which are worth mining and preserving. For example, a forming hub may indicate illegal transactions in financial network. And in user-item network, a user frequently interacts with an item could serve as a long-term feature which reflect his/her love for certain classes of products. Some models are explicitly modeled to preserve long-term features, while others implicitly learn these features when node embeddings are updated. Based on the assumption that network changes smoothly, most matrix factorization based methods adopt matrix perturbation[40] theory to update the embedding results[14, 12, 13]. For deep learning models, reusing parameters from previous time for initialization[17, 20] is one of the directly way to preserve dynamic information. However, weights reusing does not explicitly pay attention to long-term features Preservation. By maintaining sequences of nodes at different time steps, many methods[19, 32, 23, 34, 25] introduce RNN module, especially, GRU and LSTM, to particularly preserve long-term features while absorb current features in the same time. Besides, Hawkes process[31] and attention mechanism[21, 33, 26] are also introduced to model dynamic features.

- **Out-of-sample Node Embedding**

  As many classical static embedding methods are transductive algorithm[4, 48], they could not infer embeddings for out-of-sample nodes. However, most of dynamic network embedding methods, which take node addition/deletion into account, can continue model training and learn embeddings for out-of-sample nodes. Because out-of-sample nodes can be naturally treated as a part of network in next time step. Many classical static embedding methods are transductive algorithm[4, 48], which could not infer embeddings for out-of-sample nodes. However, learning node embedding on dynamic network provide a more natural way to think of out-of-sample nodes. Because those nodes can be naturally treated as a part of network in next time step. Besides, most of dynamic network embedding methods, which take node addition/deletion into account, can continue model training and learn embeddings for out-of-sample nodes. Further more, inductive learning methods on dynamic network leverage temporal features, which could bring more useful information than inductive learning methods on static network[43, 50, 56, 57].

Also, in order to improve the efficiency of incrementally model training for out-of-sample nodes, lots of tricks have been used. For example, *dynnode2vec*[20] and *NetWalk*[30] only generate random walks for changed nodes while [32] proposes *t-Batch* algorithm to accelerate training. Furthermore, *DyRep*[29] and *TGAT*[33] propose inductive algorithm that could learn representations for unseen nodes directly.

- **Prediction for Future Embedding**

  Predicting future representations is really helpful for practical applications, such as products recommendation, fraud detection. A few of models try to predict the representation of the next time step. *dyngraph2vec*[25] utilizes autoencoder structure by reducing reconstruction loss, which generates node embedding prediction for next snapshot. Besides, [32] considers that user's interest may change when they don't interact with items. Motivated by this, [32] designs a linear operator to predict the future trajectory of user in the embedding space. While *STGCN*[15] integrates gated CNN and GCN to give traffic prediction.

5. Data Sets, Evaluation and Applications

In this section, we summarize the datasets and evaluation tasks that are commonly used in developing dynamic network embedding methods. Besides, we also describe practical applications in various domains.

5.1. Datasets

We summarize commonly used dynamic network dataset with its size and granularity in table 2.

5.2. Evaluations

We briefly introduce two often used tasks in network embedding. We also list other tasks used in dynamic network embedding in table 1.

5.2.1. Node Classification

In most of graphs, a portion of nodes are assigned with labels. For example, in citation networks, a node may be labeled with its research field, whereas the labels of nodes in E-commerce may based on user's interest. Due to various factors, labels may be unknown for large fractions of nodes. The goal of node classification is to assign each node with appropriate label by learned features. Generally speaking, the process of node classification based on the network embedding could be summarized as follow: (1) Obtain low dimensional features with the network embedding approaches. (2) Train a classifier with learned embeddings to divide nodes into corresponding categories. Generally, macro-F1 and AUC (Area Under Curve) are used to evaluate the performance.

5.2.2. Link Prediction

Link prediction is also a fundamental problem in network analysis. Graphs are often incomplete in real world, e.g., links could be missing for two friends in social network. Also, researchers are interested in predicting network topology in the future. Hence, link prediction refers to predict either missing links or edges that may generate in the future. Essentially, it aims to predict whether there exists an edge between two nodes, which could be considered as a binary classification problem. Therefore, labels indicate whether there is a connection for a pair of nodes. Generally, precision @k and Mean Average Precision (MAP) are often used metrics

5.3. Practical applications

In addition to fundamental tasks listed above, dynamic network embedding methods also been adopted to practical applications across different domains. We briefly describe serval applications which handle network structure data with dynamic embedding techniques.

Table 2: Commonly used dataset

Name	Description	Nodes	Edges	Granularity
AS-733[58]	This dataset contains traffic flow exchanging in Autonomous Systems. It exhibits both the addition and deletion of the nodes and edges over time.	6,474	13,895	Daily
CollegeMsg[59]	This dataset is a directed network comprised of private messages sent on an online social network at the University of California, Irvine.	1,899	T: 59,835 S: 20,296	Seconds
DBLP[60]	This is a co-authorship network where two authors are connected if they publish at least one paper together.	317,080	1,049,866	-
wiki-talk[61]	This dataset is a network representing Wikipedia users editing each other's Talk page. A directed edge (u, v, t) means that user u edited user v's talk page at time t.	1,140,149	T: 7,833,140 S: 3,309,592	Seconds
Enron[62]	This dataset contains email messages from the Enron corpus and constructs a E-mail communication network.	143	2,347	Seconds
HEP-TH[58]	This is a citation network from the e-print arXiv., which covers papers in the period from January 1993 to April 2003 (124 months).	27770	352807	Monthly
Epinions[63]	This is a who-trust-whom online social network of a a general consumer review site Epinions.com.	75,879	508,837	-
Reddit[64]	This is a subreddit-to-subreddit hyperlink network which is extracted from the posts that create hyperlinks from one subreddit to another.	55,863	858,490	Seconds
Elliptic[65]	This is a transaction network collected from the Bitcoin blockchain. A node represents a transaction, while an edge can be viewed as a flow of Bitcoins between one transaction and the other.	203,769	234,355	49 time steps

T means the number of temporal edges while **S** means the number of static edges.

- **Anomaly Detection**

  Anomaly detection aims to identify the anomalous nodes in network, whose behaviors are different from the common nodes. And dynamic network is more likely to expose defaulter's malicious pattern. For example, in financial network, a deceiver may pretend to be a normal user in every deal but its transaction pattern through timeline would be abnormal. Early fraud detection is crucial to the minimization of loss to a financial institute and the protection of normal user rights. Anomaly detection is also important for defending network attack, detecting ticket scalper in E-commerce, etc.. Serval works have utilized dynamic network to perform detection. [34, 30, 66, 67, 68]

- **Recommender System**

  Network-based recommender systems usually take items and users as nodes and build up connections between items and items, users and users, items and users. Besides, temporal network provides fine-grained granularity of history which contains more information than static network. The dynamic network embedding methods serve as a powerful way to reduce dimension and extract items or users features, which facilitate the downstream applications. Furthermore, as the pivotal issue of recommender system is scoring the importance of an item to user, we can consider it as a link prediction problem. That is, whether we recommend a item to a user depends on whether a missing links is predicted by applied model. [24] proposes *BurstGraph* to capture unexpected bursty changes and [32] uses a coupled RNN to update embeddings for users and items. [69] completes social recommendation based on session while [36] completes next-item recommendation on temporal heterogeneous network.

- **Traffic Forecasting**

  As traffic flow can naturally be modeled as a directed network, several models of dynamic network embedding also attend to predict traffic. [15] firstly adopts graph convolutional network, [70] models traffic flows as a diffusion process and [71] focus on ride hailing demand forecasting. In addition, [72, 73, 74] are also studying traffic forecasting task.

5.4. Open Source Software

We collect a set of open source code links for dynamic network embedding methods in table 3.

6. Conclusion and Future Direction

In this survey, we conduct a comprehensive overview of the literatures in dynamic network embedding. We first provide formal definitions of commonly used data model of dynamic network: discrete model and continuous model. Then, according to the data models and corresponding methodologies, we propose a new taxonomy that organizes current dynamic network embedding methods within a novel category hierarchy, and present and compare them briefly. Some critical challenges in dynamic network embedding are also discussed. Lastly, we introduce useful dynamic network datasets and several real applications.

In the future, we believe there are at least four promising research directions for dynamic network embedding: 1) Learning embedding on heterogenous network. Heterogenous network has shown great potential to provide richer information with various node types and edge types. The static embedding on heterogenous network is well studied recently [51, 75, 76] while the research on heterogeneous dynamic graphs is just getting started[26, 36, 35, 77]. How to further explore the relationship between various types of nodes and edges in the process of network change, and to mine the unique time patterns of heterogeneous graphs is a direction worth studying. 2) Considering more general structure changes. As we summarize in Table 1, many works do not consider the deletion of node or edge. The changes of nodes and edges in the network imply a lot of information, such as may lead to the production of special subgraphs. Supporting more general structure changes would better describe the evolution of network. 3) Applying model on large-scale network. The scalability of model should also be taken into account. As the network evolves, the scale of the network may gradually grow, leading to the accumulation of snapshots in discrete model or timestamps in continuous model. How to reduce the complexity of the model to make it suitable for the ever-increasing network, while exploring more temporal information from the accumulated historical records, is a direction worthy of further study. 4) Designing task-oriented model. In addition to node classification and link prediction, there are many tasks in the field of network data mining to explore. As the network evolves, there may be dense connections between a subset of nodes, which means that a community structure is formed. How to perform representation learning on dynamic graphs for specialized tasks such as community detection is also worthy of indepth study. 5) Exploring model interpretability. Lastly, focusing on designing interpretable model could be fruitful as current approaches are limited in interpretability.

References

[1] S. T. Roweis, L. K. Saul, Nonlinear dimensionality reduction by locally linear embedding, science 290 (5500) (2000) 2323–2326.

[2] D. Wang, P. Cui, W. Zhu, Structural deep network embedding, in: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, 2016, pp. 1225–1234.

[3] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, Q. Mei, Line: Large-scale information network embedding, in: Proceedings of the 24th international conference on world wide web, 2015, pp. 1067–1077.

[4] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, 2014, pp. 701–710.

[5] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized

Table 3: A Summary of the Source Code

Structural Model	
Model	Source code
DyHNE	https://github.com/rootlu/DyHNE
TIMER	http://nrl.thumedia.org/timers-error-bounded-svd-restart-on-dynamic-networks
CTDN	https://github.com/LogicJake/CTDNE
NetWalk	https://github.com/chengw07/NetWalk
BurstGraph	https://github.com/ericZhao93/BurstGraph
DySAT	https://github.com/aravindsankar28/DySAT
STGCN	https://github.com/VeritasYin/STGCN_IJCAI-18
EvolveGCN	https://github.com/IBM/EvolveGCN
Temporal Model	
Model	Source code
Know-Evolve	https://github.com/rstriv/Know-Evolve
JODIE	http://snap.stanford.edu/jodie
DyHATR	https://github.com/skx300/DyHATR
THIGE	https://github.com/yuduo93/THIGE
Others	
Model	Source code
DynamicTriad	https://github.com/luckiezhou/DynamicTriad
TGAT	https://github.com/StatsDLMathsRecomSys/ Inductive-representation-learning-on-temporal-graphs
MMDNE	https://github.com/rootlu/MMDNE
Toolkit	
Name	Source code
DynamicGEM	https://github.com/palash1992/DynamicGEM

spectral filtering, in: Advances in neural information processing systems, 2016, pp. 3844–3852.

[6] Y. Chen, T. Qian, Relation constrained attributed network embedding, Inf. Sci. 515 (2020) 341–351. doi:10.1016/j.ins.2019.12.033.

[7] J. Chen, M. Zhong, J. Li, D. Wang, T. Qian, H. Tu, Effective deep attributed network representation learning with topology adapted smoothing, IEEE Transactions on Cyberneticsdoi:10.1109/TCYB.2021.3064092.

[8] M. E. Newman, The structure of scientific collaboration networks, Proceedings of the national academy of sciences 98 (2) (2001) 404–409.

[9] D. J. Watts, S. H. Strogatz, Collective dynamics of 'small-world'networks, nature 393 (6684) (1998) 440–442.

[10] S. M. Kazemi, R. Goel, K. Jain, I. Kobyzev, A. Sethi, P. Forsyth, P. Poupart, Representation learning for dynamic graphs: A survey., Journal of Machine Learning Research 21 (70) (2020) 1–73.

[11] Y. Xie, C. Li, B. Yu, C. Zhang, Z. Tang, A survey on dynamic network embedding, arXiv preprint arXiv:2006.08093.

[12] J. Li, H. Dani, X. Hu, J. Tang, Y. Chang, H. Liu, Attributed network embedding for learning in a dynamic environment, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pp. 387–396.

[13] Z. Zhang, P. Cui, J. Pei, X. Wang, W. Zhu, Timers: Error-bounded svd restart on dynamic networks, arXiv preprint arXiv:1711.09541.

[14] D. Zhu, P. Cui, Z. Zhang, J. Pei, W. Zhu, High-order proximity preserved embedding for dynamic networks, IEEE Transactions on Knowledge and Data

Engineering 30 (11) (2018) 2134–2144.

[15] B. Yu, H. Yin, Z. Zhu, Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting, arXiv preprint arXiv:1709.04875.

[16] L. Du, Y. Wang, G. Song, Z. Lu, J. Wang, Dynamic network embedding: An extended approach for skip-gram based network embedding, in: IJCAI, pp. 2086–2092.

[17] P. Goyal, N. Kamra, X. He, Y. Liu, Dyngem: Deep embedding method for dynamic graphs, arXiv preprint arXiv:1805.11273.

[18] J. Ma, P. Cui, W. Zhu, Depthlgp: Learning embeddings of out-of-sample nodes in dynamic networks, in: AAAI, pp. 370–377.

[19] Y. Ma, Z. Guo, Z. Ren, E. Zhao, J. Tang, D. Yin, Streaming graph neural networks, arXiv preprint arXiv:1810.10627.

[20] S. Mahdavi, S. Khoshraftar, A. An, dynnode2vec: Scalable dynamic network embedding, in: 2018 IEEE International Conference on Big Data (Big Data), IEEE, pp. 3762–3765.

[21] A. Sankar, Y. Wu, L. Gou, W. Zhang, H. Yang, Dynamic graph representation learning via self-attention networks, arXiv preprint arXiv:1812.09430.

[22] L.-k. Zhou, Y. Yang, X. Ren, F. Wu, Y. Zhuang, Dynamic network embedding by modeling triadic closure process, in: AAAI, pp. 571–578.

[23] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, C. E. Leisersen, Evolvegcn: Evolving graph convolutional networks for dynamic graphs, arXiv preprint arXiv:1902.10191.

[24] Y. Zhao, X. Wang, H. Yang, L. Song, J. Tang, Large scale evolving graphs with burst detection, in: IJCAI, pp. 4412–4418.

[25] P. Goyal, S. R. Chhetri, A. Canedo, dyngraph2vec: Capturing network dynamics using dynamic graph representation learning, Knowledge-Based Systems 187 (2020) 104816.

[26] L. Yang, Z. Xiao, W. Jiang, Y. Wei, Y. Hu, H. Wang, Dynamic heterogeneous graph embedding using hierarchical attentions, in: European Conference on Information Retrieval, Springer, pp. 425–432.

[27] G. H. Nguyen, J. B. Lee, R. A. Rossi, N. K. Ahmed, E. Koh, S. Kim, Continuous-time dynamic network embeddings, in: Companion Proceedings of the The Web Conference 2018, pp. 969–976.

[28] R. Trivedi, H. Dai, Y. Wang, L. Song, Know-evolve: Deep temporal reasoning for dynamic knowledge graphs, arXiv preprint arXiv:1705.05742.

[29] R. Trivedi, M. Farajtabar, P. Biswal, H. Zha, Dyrep: Learning representations over dynamic graphs, in: International Conference on Learning Representations.

[30] W. Yu, W. Cheng, C. C. Aggarwal, K. Zhang, H. Chen, W. Wang, Netwalk: A flexible deep embedding approach for anomaly detection in dynamic networks, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2672–2681.

[31] Y. Zuo, G. Liu, H. Lin, J. Guo, X. Hu, J. Wu, Embedding temporal network via neighborhood formation, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2857–2866.

[32] S. Kumar, X. Zhang, J. Leskovec, Predicting dynamic embedding trajectory in temporal interaction networks, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1269–1278.

[33] D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, K. Achan, Inductive representation learning on temporal graphs, arXiv preprint arXiv:2002.07962.

[34] L. Zheng, Z. Li, J. Li, Z. Li, J. Gao, Addgraph: Anomaly detection in dynamic graph using attention-based temporal gcn, in: IJCAI, pp. 4419–4425.

[35] H. Xue, L. Yang, W. Jiang, Y. Wei, Y. Hu, Y. Lin, Modeling dynamic heterogeneous network for link prediction using hierarchical attention with temporal rnn, arXiv preprint arXiv:2004.01024.

[36] Y. Ji, M. Yin, Y. Fang, H. Yang, X. Wang, T. Jia, C. Shi, Temporal heterogeneous interaction graph embedding for next-item recommendation.

[37] Y. Lu, X. Wang, C. Shi, P. S. Yu, Y. Ye, Temporal network embedding with micro-and macro-dynamics, in: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, 2019, pp. 469–478.

[38] D. Cai, X. He, J. Han, T. S. Huang, Graph regularized nonnegative matrix factorization for data representation, IEEE transactions on pattern analysis and machine intelligence 33 (8) (2010) 1548–1560.

[39] S. Cao, W. Lu, Q. Xu, Grarep: Learning graph representations with global structural information, in: Proceedings of the 24th ACM international on conference on information and knowledge management, 2015, pp. 891–900.

[40] G. W. Stewart, Matrix perturbation theory.

[41] X. Wang, Y. Lu, C. Shi, R. Wang, P. Cui, S. Mou, Dynamic heterogeneous information network embedding with meta-path based proximity, IEEE Transactions on Knowledge and Data Engineering.

[42] H. Bourlard, Y. Kamp, Auto-association by multilayer perceptrons and singular value decomposition, Biological cybernetics 59 (4-5) (1988) 291–294.

[43] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, in: I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems 30, Curran Associates, Inc., 2017, pp. 1024–1034.

[44] D. P. Kingma, M. Welling, Auto-encoding variational bayes, arXiv preprint arXiv:1312.6114.

[45] Y. Goldberg, O. Levy, word2vec explained: deriving

mikolov et al.'s negative-sampling word-embedding method, arXiv preprint arXiv:1402.3722.

[46] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, 2016, pp. 855–864.

[47] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and locally connected networks on graphs, arXiv preprint arXiv:1312.6203.

[48] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, arXiv preprint arXiv:1609.02907.

[49] M. Niepert, M. Ahmed, K. Kutzkov, Learning convolutional neural networks for graphs, in: International conference on machine learning, 2016, pp. 2014–2023.

[50] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, arXiv preprint arXiv:1710.10903.

[51] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, P. S. Yu, Heterogeneous graph attention network, in: The World Wide Web Conference, 2019, pp. 2022–2032.

[52] Q. Cui, S. Wu, Y. Huang, L. Wang, A hierarchical contextual attention-based network for sequential recommendation, Neurocomputing 358 (2019) 141–149.

[53] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning representations by back-propagating errors, nature 323 (6088) (1986) 533–536.

[54] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, arXiv preprint arXiv:1412.3555.

[55] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (8) (1997) 1735–1780.

[56] R. A. Rossi, R. Zhou, N. K. Ahmed, Deep inductive network representation learning, in: Companion Proceedings of the The Web Conference 2018, 2018, pp. 953–960.

[57] Y. Bai, H. Ding, Y. Qiao, A. Marinovic, K. Gu, T. Chen, Y. Sun, W. Wang, Unsupervised inductive graph-level representation learning via graph-graph proximity, arXiv preprint arXiv:1904.01098.

[58] J. Leskovec, J. Kleinberg, C. Faloutsos, Graphs over time: densification laws, shrinking diameters and possible explanations, in: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining, 2005, pp. 177–187.

[59] P. Panzarasa, T. Opsahl, K. M. Carley, Patterns and dynamics of users' behavior and interaction: Network analysis of an online community, Journal of the American Society for Information Science and Technology 60 (5) (2009) 911–932.

[60] J. Yang, J. Leskovec, Defining and evaluating network communities based on ground-truth, Knowledge and Information Systems 42 (1) (2015) 181–213.

[61] A. Paranjape, A. R. Benson, J. Leskovec, Motifs in temporal networks, in: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, 2017, pp. 601–610.

[62] B. Klimt, Y. Yang, Introducing the enron corpus., in: CEAS, 2004.

[63] M. Richardson, R. Agrawal, P. Domingos, Trust management for the semantic web, in: International semantic Web conference, Springer, 2003, pp. 351–368.

[64] S. Kumar, W. L. Hamilton, J. Leskovec, D. Jurafsky, Community interaction and conflict on the web, in: Proceedings of the 2018 World Wide Web Conference, 2018, pp. 933–943.

[65] M. Weber, G. Domeniconi, J. Chen, D. K. I. Weidele, C. Bellei, T. Robinson, C. E. Leiserson, Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics, arXiv preprint arXiv:1908.02591.

[66] C. Yang, L. Zhou, H. Wen, Z. Zhou, Y. Wu, H-vgrae: A hierarchical stochastic spatial-temporal embedding method for robust anomaly detection in dynamic networks, arXiv preprint arXiv:2007.06903.

[67] M. Garchery, M. Granitzer, Adsage: Anomaly detection in sequences of attributed graph edges applied to insider threat detection at fine-grained level, arXiv preprint arXiv:2007.06985.

[68] S. Lagraa, K. Amrouche, H. Seba, et al., A simple graph embedding for anomaly detection in a stream of heterogeneous labeled graphs, Pattern Recognition (2020) 107746.

[69] W. Song, Z. Xiao, Y. Wang, L. Charlin, M. Zhang, J. Tang, Session-based social recommendation via dynamic graph attention networks, in: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, 2019, pp. 555–563.

[70] Y. Li, R. Yu, C. Shahabi, Y. Liu, Diffusion convolutional recurrent neural network: Data-driven traffic forecasting, arXiv preprint arXiv:1707.01926.

[71] X. Geng, Y. Li, L. Wang, L. Zhang, Q. Yang, J. Ye, Y. Liu, Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 3656–3663.

[72] H. Lu, D. Huang, Y. Song, D. Jiang, T. Zhou, J. Qin, St-trafficnet: A spatial-temporal deep learning network for traffic forecasting, Electronics 9 (9) (2020) 1474.

[73] C. Song, Y. Lin, S. Guo, H. Wan, Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, 2020, pp. 914–921.

[74] H. Peng, H. Wang, B. Du, M. Z. A. Bhuiyan, H. Ma, J. Liu, L. Wang, Z. Yang, L. Du, S. Wang, et al., Spatial temporal incidence dynamic graph neural networks for traffic flow forecasting, Information Sciences 521 (2020) 277–290.

[75] C. Zhang, D. Song, C. Huang, A. Swami, N. V. Chawla, Heterogeneous graph neural network, in:

Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 793–803.

[76] S. Fan, J. Zhu, X. Han, C. Shi, L. Hu, B. Ma, Y. Li, Metapath-guided heterogeneous graph neural network for intent recommendation, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 2478–2486.

[77] R. Bian, Y. S. Koh, G. Dobbie, A. Divoli, Network embedding and change modeling in dynamic heterogeneous networks, in: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2019, pp. 861–864.