Learning and Updating Node Embedding on Dynamic Heterogeneous Information Network

Yuanzhen Xie
School of Computer Science and Engineering, Sun Yat-sen University
xieyzh3@mail2.sysu.edu.cn

Zijing Ou
School of Computer Science and Engineering, Sun Yat-sen University
ouzj@mail2.sysu.edu.cn

Liang Chen*
School of Computer Science and Engineering, Sun Yat-sen University
chenliang6@mail.sysu.edu.cn

Yang Liu
School of Computer Science and Engineering, Sun Yat-sen University
liuy296@mail2.sysu.edu.cn

Kun Xu
School of Computer Science and Engineering, Sun Yat-sen University
xukun6@mail2.sysu.edu.cn

Carl Yang
Emory University
j.carlyang@emory.edu

Zibin Zheng
School of Computer Science and Engineering, Sun Yat-sen University
zhzibin@mail.sysu.edu.cn

ABSTRACT

Heterogeneous information networks consist of multiple types of edges and nodes, which have a strong ability to represent the rich semantics underpinning network structures. Recently, the dynamics of networks has been studied in many tasks such as social media analysis and recommender systems. However, existing methods mainly focus on the static networks or dynamic homogeneous networks, which are incapable or inefficient in modeling dynamic heterogeneous information networks. In this paper, we propose a method named Dynamic Heterogeneous Information Network Embedding (DyHINE), which can update embeddings when the network evolves. The method contains two key designs: (1) A dynamic time-series embedding module which employs a hierarchical attention mechanism to aggregate neighbor features and temporal random walks to capture dynamic interactions; (2) An online real-time updating module which efficiently updates the computed embeddings via a dynamic operator. Experiments on three real-world datasets demonstrate the effectiveness of our model compared with state-of-the-art methods on the task of temporal link prediction.

CCS CONCEPTS

• **Information systems** → **Social networks**; • **Networks** → *Network dynamics*; • **Computer systems organization** → Neural networks.

*Corresponding author.

KEYWORDS

dynamic network embedding, heterogeneous network, hierarchical attention mechanism

ACM Reference Format:
Yuanzhen Xie, Zijing Ou, Liang Chen, Yang Liu, Kun Xu, Carl Yang, and Zibin Zheng. 2021. Learning and Updating Node Embedding on Dynamic Heterogeneous Information Network. In *Proceedings of the Fourteenth ACM International Conference on Web Search and Data Mining (WSDM '21), March 8–12, 2021, Virtual Event, Israel.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3437963.3441745

1 INTRODUCTION

Many real-world applications are built on networks [31] such as biomolecular networks, social networks, and information networks. In order to better represent networks, many researchers study various kinds of network embedding methods, which aim to learn vertex representation in a low-dimensional space while preserving the network structure. These embeddings can benefit many downstream network learning tasks such as recommendation [21, 29], link prediction [13, 26], node classification [33], and community detection [28]. However, most real-world networks like e-commerce or social network are heterogeneous and they evolve constantly. Intuitively, the use of temporal information is helpful to learn network embedding, and rapid embedding updates can serve many downstream tasks in real-time. Therefore, dynamic heterogeneous network embedding becomes an urgent problem worth studying.

In the past years, lots of network embedding methods [2, 5, 18, 21, 24, 27, 34, 37] have emerged. Based on whether or not to consider temporal information, we can classify them into static and dynamic embedding methods. Static methods [2, 5, 18, 21, 24, 27] have already covered both homogeneous and heterogeneous networks, but they do not consider temporal information and cannot learn the rich features contained in the network evolution. Dynamic methods [1, 6, 12, 15, 34, 36, 37] learn these features based on embedding snapshots or using temporal evolution models, but none of them can be directly applied to heterogeneous networks.

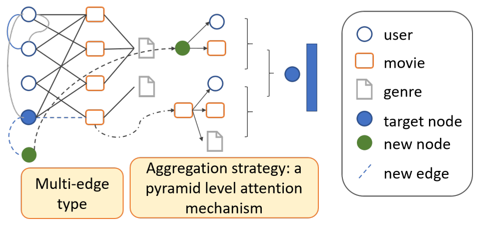


Figure 1: Toy example of a dynamic heterogeneous network (Different shapes denote different node types.)

In this paper, we mainly focus on dynamic heterogeneous network representation learning. With the rapid development of technology, real-world networks are getting larger and more complex. As shown in Fig.1, there are various kinds of nodes and edges and they change over time. Existing methods considering static information or homogeneous network are not enough to obtain good results in dynamic heterogeneous networks as shown in the experiment. Besides, several problems emerge like that most of the current dynamic methods cost a lot when updating, since they need to recompute the embedding of all nodes. In conclusion, the challenging issues are how to learn the structural information of dynamic heterogeneous networks and how to quickly update the embedding when the network changes. Naturally, we have a conventional approach to deal with it. That is, we can apply static network embedding algorithms to the network snapshot of each timestamp and just adjust some affected nodes. However, the major drawback of this intuitive idea is that generating embedding in each timestamp consumes a lot of time and resources.

To sum up, the development of networks imposes new challenges on network embedding methods.

- **(C1) Heterogeneity.** The heterogeneous information network has different node types, and each node may have multiple edge types. Different types of edges and nodes have different effects on the target node. Therefore, an appropriate aggregation strategy is needed to find to aggregate this structural information.
- **(C2) Dynamics.** Networks evolve over time. When learning node representation, how to maintain temporal network structure information, such as high-order proximity and structural information, is a problem worth pondering over.
- **(C3) Efficiency.** Because large-scale networks may have hundreds of millions of nodes and edges, an efficient online updating method is needed to update network embedding in real-time.

To address the three major challenges presented, we design a novel unsupervised dynamic heterogeneous network embedding method named Dynamic Heterogeneous Information Network Embedding (DyHINE). It consists of two core models, named dynamic time-series embedding model and online real-time update model respectively. The dynamic time-series embedding model contains two key designs. One is to learn dynamic information from node sequences obtained by temporal random walk. The other is to design a hierarchical attention mechanism to learn the information on heterogeneous networks. The hierarchical attention mechanism is to aggregate the neighbor node and different edge type information. Specifically, for the target node, the first attention layer is used to aggregate the neighbor node embedding and the embedding of edges of the same type around it; the second attention layer aggregates edges by types, and then uses a fusion method to further aggregate all edges. In the online real-time update model, we use time-based operators to learn the changing node information, and then only adjust and update the embedding of the affected nodes to save training time. At last, an additional objective function is designed to constrain the learned embedding. Since most networks do not have node labels, we focus on unsupervised network embedding.

To summarize, the main **contributions** of DyHINE are as follows:

- An unsupervised network embedding method based on hierarchical attention mechanism and temporal random walk is proposed to learn the embedding of nodes in heterogeneous information networks.
- The online real-time update method is proposed to study nodes' information dynamically and reduce time consumption. Avoiding updating the whole network, the incremental learning method is proposed to update the representation of nodes that are affected by network changes.
- We conduct experiments on three datasets and the results demonstrate the superior performance of DyHINE over state-of-the-art baselines on the task of temporal link prediction.

2 PROBLEM DEFINITION

In this part, we first introduce key concepts including heterogeneous networks, dynamic heterogeneous networks and dynamic heterogeneous network embedding, then we formally define our problem.

Definition 1. *(Heterogeneous Network). A heterogeneous network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a form of graph where $\mathcal{V}$ and $\mathcal{E}$ represent the sets of nodes and edges, respectively. It is associated with a node type mapping function $\mu_1 : \mathcal{V} \rightarrow A$ and an edge type mapping function $\mu_2 : \mathcal{E} \rightarrow R$, where A and R denote the set of all nodes and edges types, respectively. Each node $v \in \mathcal{V}$ belongs to a particular node type. Similarly, each edge $e \in \mathcal{E}$ is categorized into a specific edge type. If $|A| + |R| > 2$, it is called heterogeneous network.*

Definition 2. *(Dynamic Heterogeneous Network). A dynamic heterogeneous network is a graph $\mathcal{G} = \{\mathcal{G}^1, \ldots, \mathcal{G}^t\}$ with a series of network snapshots within a time interval. $\mathcal{G}^t = (\mathcal{V}^t, \mathcal{E}^t, \mathcal{W}^t)$ is a directed network snapshot at time t, where $\mathcal{V}^t$, $\mathcal{E}^t$ and $\mathcal{W}^t$ represent the sets of nodes, edges, the edges type at time t, respectively.*

Definition 3. *(Dynamic Heterogeneous Network Embedding). Given a dynamic heterogeneous networks $\mathcal{G} = \{\mathcal{G}^1, \ldots, \mathcal{G}^t\}$, it aims to learn a mapping function $\mu_t : v_i \rightarrow \mathcal{R}^d$ for each timestamp t, where d is a positive integer indicating the number of embedding dimensions. The objective of the function μ_t is to preserve the similarity between v_i and v_j on both the network structure at timestamp t and predict their tendencies to develop relationships with others in the future.*

We are committed to solving the problem of node representation learning in dynamic heterogeneous information network and give an efficient method to update embedding during a short time. Dynamic heterogeneous network embedding can be divided into two parts: one is base embedding, the other is the embedding of adjusting the changed nodes within a certain time. After given the above definition, our problem can be defined as:

Problem 1. *Given $\mathcal{G} = \{\mathcal{G}^1, \dots, \mathcal{G}^t\}$ and base embedding, the **input** is the changed set of edges $\mathcal{E}_{ch} = \{(n1, n2, etype, 1), (n2, n3, etype, 0), \dots\}$, where 1 and 0 represent the operation of creating and deleting edges, respectively. The **output** is node representation $G^{t+\Delta}$ capturing the evolutionary trajectory of the graph, where Δ denotes time variation.*

3 PROPOSED METHOD

In this section, we first explain the design motivation and the proposed model architecture, namely Dynamic Heterogeneous Information Network Embedding (DyHINE) method. Then, we describe the corresponding model structure which aims to solve the three challenges in detail and give the objective optimization function.

3.1 Model Overview

The network structure is continuously changing over time in reality. In order to learn more purposeful node representation, the evolutionary information needs to be employed. Furthermore, node embedding should be updated in real-time to provide superior downstream services. For example, in the e-commerce shopping scene, users' preferences should be updated instantly after user clicking on a product to improve the recommendation performance.

To meet different requirements, we design two models: dynamic time-series embedding model and online real-time update model. In the former, we focus on the heterogeneity and dynamics of the network to mine rich features. In the latter, the few new nodes change their features in a short time is assumed to ensure efficiency, so only the affected nodes are updated. In summary, the purpose of the dynamic time-series embedding model is to learn the time-varying features of nodes in heterogeneous networks, while online real-time update model is to meet the need for updating embedding in a short time.

Our idea of designing models is dedicated to solving the three major challenges presented. **Firstly**, network has heterogeneity as shown in Fig.1. The users rate the film and television works, and the film and television works also have their category. There are some different nodes and relationship links between nodes, such as indicating that users like films and who is the star of a film. In this paper, the features of nodes and edges are represented by vectors, which can be called node or edge embedding. Node embedding is represented by $G \in R^{n \times d}$ (where n represents the number of nodes and d represents the dimension of embedding), and edge embedding is represented by $E \in R^{n \times s \times e}$ (where e is the edge embedding size and s denotes the number of edge types). Nodes are susceptible to the changes of their neighbors, and different types of nodes or edges have different importance to the target node. Therefore, we design a hierarchical attention mechanism to learn the diversity of edges and nodes to solve **C1** (as described in section 3.2). **Secondly**, the network is constantly evolving, and the features of nodes are constantly changing. To solve **C2**, we adopt temporal random walk and design the corresponding loss function. (Please see section 3.3 for details.) **Thirdly**, according to the actual situation, the number of changed edges and nodes in the network is extremely small in a short period, thus the embedding should not change a lot and rerunning the whole network is unnecessary. Especially for large-scale networks, the rerun strategy is not appropriate when new nodes and edges are coming. To solve **C3**, an online update model is designed. If some nodes are changed, only the features of affected nodes (edge type, neighbor embedding) are used to update their embedding. Experiments show that it can also achieve good results. In the loss design, our goal is to find the embedding of each node at time t based on the unsupervised model. Inspired by Deepwalk [18], we used skip-gram as the basic model to learn node embedding.

3.2 Heterogeneous Network Embedding Learning

To solve **C1**, we design the hierarchical attention mechanism which can be viewed as a pyramid level attention mechanism as shown in Fig.2 (the figure is shown horizontally). We select the network snapshot diagram at the current time for operation and our model can also be used for continuous-time update learning. The first layer firstly aggregates different neighbor edge and node embedding (the neighbor and edge relations formed before the current time) and then aggregates the neighbor node and edge embedding in the second layer.

Node aggregation. Learning the information of neighbor nodes in heterogeneous networks, we have some intuitive ways to integrate the neighbor nodes. For example, the final fusion result can be calculated by the mean-pooling, max-pooling or min-pooling method with multiple embedding matrix. But these methods fail to consider the contribution of different nodes. In reality, users can interact with other users or rate movies. They should have different contributions or influences for users. Thus we use the attention mechanism to learn the information of neighbor nodes.

An obvious fact is that the number of node neighbors could be extremely large without giving a limitation on neighborhood distance. Thus, we set the number of neighbors as a super parameter (we can choose to use the whole neighbors or a fixed number of neighbors in the experiment) and fuse the neighbor information into the target node. If the target node v has m neighbor nodes in the network snapshot at time t, denoting as $\{vn_1, vn_2, ..., vn_m\}$, the embedding combination of these neighbor nodes is set as $N_v \in R^{d \times m}$, with

$$N_v = [G_{vn_1}; G_{vn_2}; ...; G_{vn_m}], \tag{1}$$

where, G_{vn_m} denotes one of the neighbors embedding of node v. To efficiently capture the neighbors' influence, we exploit attention mechanism, and the gravity coefficient of different neighbors is:

$$\beta_v = softmax(W_{n1}^T tanh(W_{n2} N_v))^T, \tag{2}$$

where, $W_{n1} \in R^d$ and $W_{n2} \in R^{d \times d}$ are the parameters of learning neighbors information for node v.

Considering the influence of the previous moment and the change of embedding at the current moment, the neighbor node embedding after aggregation is proposed as follows:

$$G_v^{t'} = maxpooling(N_v^t \beta_v^t, G_v^t), \tag{3}$$

where, G_v denotes the representation of node v, t represents the timestamp.

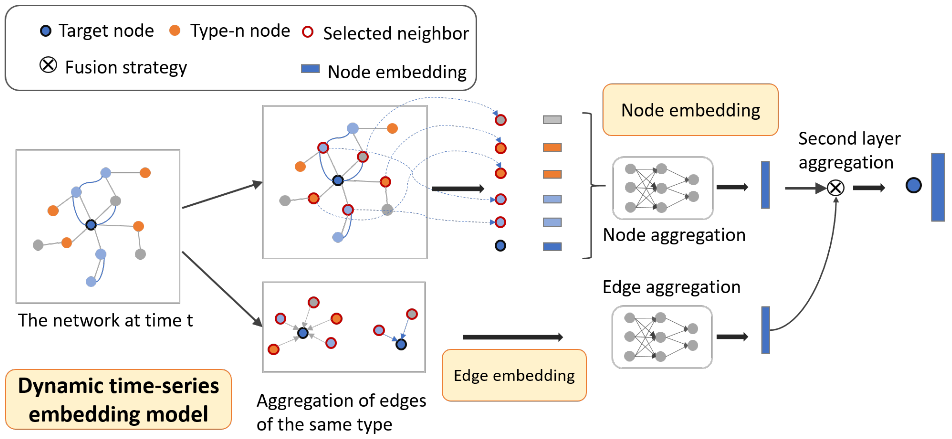


Figure 2: Overview of the dynamic time-series embedding model of DyHINE method: the network at time t is firstly obtained for the target node, then representations of different types of edges and neighbor nodes are learned by neural networks, and the final representations are generated via aggregating the edge and neighbor nodes representations.

Edge aggregation. Through the aggregation of neighbor nodes, the structural characteristics of the network have been roughly learned. And edge features are needed to learn to solve network heterogeneity better, so edge embedding $E \in R^{n \times s \times e}$ is introduced. In the edge aggregation calculation, only the edge embedding with first-order neighbors is calculated.

Similarly, the influence of different edge types should be different. For example, in the friend recommendation scenario, the contribution of friends is greater than group relationships and the contribution of the best friend is greater than a normal friend. For this reason, a novel dual-level attention mechanism is designed to learn about different types of edges.

Firstly, we apply type-level attention to learn the representation $P_{v,r} \in R^{e}$ of the same type edges of neighboring nodes:

$$P_{v,r} = aggregator(E_{j,r}, j \in \mathcal{N}_{v,r}), \tag{4}$$

where, v represents target node, $\mathcal{N}_{v,r}$ is the neighbors of node v on edge type r, and the aggregator function is the same as attention mechanism used in Formula (2) and (3).

Then, we design the edge-level attention to capture the importance of different type edges in the target node. Formally, different type of edge representations are aggregated first:

$$P_v = [P_{v,1}; P_{v,2}; ...; P_{v,s}], \tag{5}$$

where s is the size of edge type. Then edge-level attention weights are computed with softmax function:

$$\alpha_v = softmax(W_{e1}^T tanh(W_{e2} P_v))^T, \tag{6}$$

where, $W_{e1} \in R^{d}$ and $W_{e2} \in R^{d \times e}$ are the parameters of learning neighbors information for node v. Finally, we incorporate dual-level attention mechanism into edge aggregation with the following layer-wise propagation rule:

$$H_v^{t'} = MP_v \alpha_v, \tag{7}$$

where, $M \in R^{d \times e}$ is the transformation matrix for the unification of the edge and node embedding.

Second layer aggregation. In the second level of aggregation, we focus on the aggregation of edge embedding and node embedding. One could also have a simple aggregation operator over them to model interactions between node and edge, but we find that this resulted in worse performance. Instead we obtain improvements by a gate mechanism to distill the complementary information from $G_v^{t'}$ and $En_v^{t'}$. Specifically, the gate is designed as:

$$g_v = 1 - \sigma(W_{g1} H_v^{t'} + b_{g1}) \odot G_v^{t'}, \tag{8}$$

where, $\sigma(.)$ is the sigmoid function, $\odot$ denotes the element-wise multiplication, $W_{g1} \in R^{d \times d}$ and $b_{g1} \in R^{d}$ are learnable parameters used to align edge embedding to node embedding space. After that, cooperated with a new trainable matrix $W_{g2} \in R^{d \times d}$ and bias $b_{g2} \in R^{d}$, the final aggregation is computed as:

$$G_v^{t+1} = ((W_{g2} H_v^{t'} + b_{g2}) \odot G_v^{t'}) \odot g_v, \tag{9}$$

where, $t+1$ represents the next timestamp.

3.3 Dynamic Network Embedding Learning

As the network evolves over time, the features of nodes are constantly enriched and changed. Time is a critical factor in dynamic learning because the order of relationship formed by nodes and edges contains rich semantic. For example, in the network formed by item recommendation, the order of forming edges between users and commodities not only reflects the change of users' preferences, but also reflects the possible combination of commodities. Therefore, some strategies are designed to capture timing information.

At first, based on the loss function of original skip-gram model, time dimension $||G^t - G^{t-\Delta}||_2$ is added. The motivation comes from the fact that there are few changes during a short time. Secondly, the temporal random walk strategy is adopted. That is, it walks out of the sequence according to the time sequence relationship. For example, sequence $a \longrightarrow b \longrightarrow c$ indicates that node b builds up relation with a before c. And the length of the random walk is fixed in our model. In addition, time-based batch is used to sort

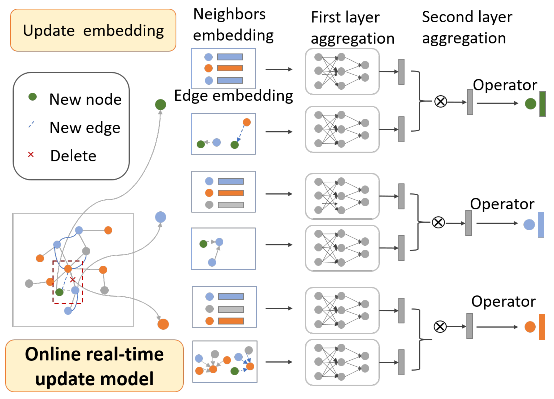


Figure 3: Overview of the online real-time update model of DyHINE method: it first obtains the change edges and finds the influenced nodes, then it adjusts their embeddings.

the training and testing datasets in ascending order of time. Time-based batch refers to dividing batches for learning according to time points.

3.4 Efficient Embedding Updating

The learning of large networks faces the problem of computational complexity. In some practical applications, such as real-time monitoring of abnormal behavior, real-time network changes need to be quickly learned to serve downstream tasks. Generally speaking, the nodes affected by network changes in a short time may be few in a large-scale network. Updating all nodes wastes a lot of time. Therefore, some affected nodes are found to update to reduce time consumption. For these reasons, we design the model shown in Fig.3. The model dynamically updates the representation of nodes when edge relations of nodes are established or deleted. In the training, we use the idea of flow to learn data one by one. To meet the requirement of the rapid update, embedding is divided into two parts: past embedding and changing embedding. The changing embedding uses the operator w to learn directly.

When a new node is formed, it can be learned when it establishes new edge connections. Before that, just the random or mean initialization is done to learn its embedding. Based on this point, we first put forward the updating process of the node in the case that it has new edges establishment. The embedding of the new node can be updated with the edge establishment method when it has established the new edge relationship.

Edge change. There are many scenes of changing edges such as establishing a new chat relationship, users rating a movie, students publishing a paper, and so on. Given the set of changed edges $\mathcal{E}_c = \{(n1, n2, etype, 1), (n2, n3, etype, 0), \dots\}$, we find the adjacent affected nodes and only update them by using operator $w \in R^d$ to reduce time complexity. $w = W_o \times \Delta$. W_o is initialized by Gaussian distribution. Assuming that v is one of the affected nodes, inspired by [12], the following function is used to study the embedding of node v:

$$G_{v,d}^{t+\Delta} = (1 + w) \odot G_v^t, \tag{10}$$

where , G_v^t denotes the embedding learning by dynamic time-series model, G_{*d}^{t+1} represents dynamic node embedding in online real-time update model and $w \in R^d$ denotes the operator of node v. There are no deleted edges in the experiment, and a feasible theoretical method is proposed here.

Node change. On the Internet, users may register and delete accounts, which brings changes to the network structure. A similar solution to edge processing is adopted to ensure that node embedding can be updated in real-time. When a new node comes, it will first initialize its embedding with mean value of all nodes. Then when it establishes a relationship with other nodes, the edge increase strategy will be used to update the network. When the node is deleted, we will find the affected node, and then update these nodes using the operator. The formula is the same as Formula 10.

3.5 Objective and Model Training

To perform dynamic heterogeneous graph representation learning, we define the following objective function:

$$E = argmax \prod_{v \in \mathcal{V}} \prod_{t \in T_{\mathcal{V}}} \prod_{v_c \in N_v^t} p(v_c|v;\theta) \tag{11}$$

where, N_v^t is the set of neighbors of node v in the random walk of the graph at a timestamp, θ denotes all the model parameters. Following Node2vec [10], we use softmax function to define the conditional probability $p(v_c|v;\theta)$ as:

$$p(v_c|v;\theta) = \frac{exp(G_{v_c} \cdot G_v)}{\sum_{k \in \mathcal{V}_t} exp(G_{v_k} \cdot G_v)} \tag{12}$$

where, G_v is the output node embedding from the proposed model. Similar to [16], we use the negative sampling technique to approximate the function $p(v_c|v;\theta)$ as:

$$p(v_c|v;\theta) = log\sigma(G_{v_c} \cdot G_v) + \sum_{l=1}^{L} \Xi_{v_k \sim P_t(v_k)} log\sigma(-G_{v_k} \cdot G_v) \tag{13}$$

where, L denotes the negative sample size and v_k is randomly selected from a noise distribution $P_t \propto d_{v_k}^{3/4}$, with d_{v_k} denoting the out-degree of v_k. In the selection of positive and negative samples, a pair of positive and negative samples are selected for each central node.

The objective function in the dynamic time-series embedding model is described above, and in the online real-time update model, considering the change is slight, the following one is added to its' objective function:

$$E' = argmin \sum_{t \in T} \sum_{v \in \mathcal{V}_{ch}} ||G_v^t - G_v^{t-\Delta}||_2 \tag{14}$$

where, G^t denotes the dynamic embedding in time t while $G^{t-\Delta}$ denotes the dynamic embedding before time t and $\mathcal{V}_{ch}$ denotes the nodes whose edges have changed.

Finally, the objective function of DyHINE model is:

$$E_d = argmin(\sum_{t \in T} \sum_{v \in \mathcal{V}_{ch}} ||G_v^t - G_v^{t-\Delta}||_2 - \prod_{v \in \mathcal{V}_{ch}} \prod_{t \in T_{\mathcal{V}_{ch}}} \prod_{v_c \in N_v^t} p(v_c|v;\theta)) \tag{15}$$

Table 1: Statistics of Datasets.

Datasets	#nodes	#edges	#node-type	#edge-type	density
Twitter	27711	40000	1	3	0.0052%
Movielens	2643	182799	3	3	2.62%
Amazon	11930	163951	1	4	0.12%

4 EXPERIMENT

In this section, we conduct extensive experiments aiming to answer the following research questions:

- **RQ1** How does DyHINE perform vs. state-of-the-art baselines on the temporal link prediction task?
- **RQ2** What are the effects of different neighbors and surrounding edges features on the target node?
- **RQ3** How efficient is DyHINE compared with other dynamic methods?
- **RQ4** How does model parameters like embedding size affect the model?

4.1 Datasets

Three public datasets are selected to complete the temporal link prediction task. The statistics of datasets are summarized in Table 1. In the process of network construction, several relationships are selected to form heterogeneous networks. The data is sorted in chronological order. The data before a certain point in time is used as the training set, and the last data is used as the testing set. In the experiment, we also divide the training set into about 9:1 according to timestamp. The first nine parts are learned according to the proposed dynamic time-series embedding model, and the last part is learned by streaming according to the online real-time update model. There are some new nodes in the last part that have not previously appeared. The link relationship of new nodes in the testing set needs to be predicted. For other methods, the entire training set is trained without involving new nodes. In the application of the formula, the first nine parts use Formula 9, while the last one part use Formula 10. A brief introduction to the data set is given below:

Twitter. Twitter dataset (Higgs) comes from social applications. It consists of four directional relationships, namely forwarding, replying, and mentioning relationships between more than 450,000 Twitter users. The data format is "userA, userB, timestamp, interaction" and each edge type is balanced. In our experiment, all edge types are tested.

Movielens. Movielens dataset has three node types that consist of four types of edges with timestamp. We select users' ratings for movies, movies' genre, movies and users' relationship. The distribution of different edge types is not uniform, and the number of one edge type is much higher than the other. Only one main edge type(movies and users) are tested, which can be seen as a recommendation question.

Amazon. Amazon dataset in our experiment is derived from [3], which has four types of edges without timestamp. It includes product metadata and links between products. As Movielens dataset, there is only one edge type in its' testing set.

4.2 Experimental Setup

Our model code is completed with Pytorch. To reduce experimental adjustment parameters and better compare the effectiveness of each method, for all embedding methods, some parameters are fixed if they have: The length of the walkers: 10; The window size: 5; The number of neighbor samples: 10; The size of negative samples: 5. The node embedding size of our model is 64, and other methods select it by tuning.

Baseline methods. The proposed method is compared with several state-of-the-art methods using public codes or our implementations. In terms of comparison method selection, we choose three kinds of network embedding methods, categorized as static homogeneous network embedding methods (Deepwalk, LINE, Node2vec), static heterogeneous network embedding methods (GATNE), and dynamic network embedding methods (CTDNE, DynamicTriad, tNodeEmbed, DynAERNN). In the test method, we first use different methods to generate the node vector representation, and then use the temporal link task to test the generated results.

- **Deepwalk.** Deepwalk [18] generalizes word embedding and uses truncated random walks to learn potential representations of networks.
- **LINE.** LINE [24] designs an optimized objective function to preserve first-order and second-order approximations for learning network representations.
- **Node2vec.** Node2vec [10] designs a biased random walk procedure for static network and it introduces two parameters to control the random walk process.
- **GATNE.** GATNE [3] formalizes the embedded learning problem of heterogeneous networks with multiple attributes and proposes a unified framework to solve it. The framework supports both transductive and inductive learning.
- **CTDNE.** CTDNE [17] generalizes random walk-based embedding methods to a dynamic scenario. Several effective meta-path selection strategies are proposed to capture temporal properties. Specifically, the unbias strategy is used in this paper.
- **DynamicTriad.** DynamicTriad [36] is incapable of distinguishing different types of objects, which could result in incorrect meta-paths being adopted for modeling the triad closure process.
- **tNodeEmbed.** tNodeEmbed [22] utilizes a joint loss function to optimize specific given tasks, which simultaneously obtains a temporal embedding of a node by learning to combine historical temporal embeddings.
- **DynAERNN.** DynAERNN [9] extends the autoencoders to incrementally generate embedding of a dynamic graph by using Recurrent Neural Network to capture the temporal correlations.

4.3 The Performance on Temporal Link Prediction (RQ1)

To answer **RQ1**, we set up some experiments to estimate the effectiveness of our proposed model on temporal link prediction task.

Temporal link prediction. Link prediction is a common task in academia and industry. In academia, it is often used to estimate the advantages and disadvantages of different network embedding methods. The different is that temporal link prediction considers the time, which can be viewed as predicting future connections. In this experiment, we hide the later part of the edge as the testing set

Table 2: Perfromance comparison of different methods on three datasets for temporal link prediction task.

Method	Twitter			Movielens			Amazon		
	ROC-AUC	F1	PR-AUC	ROC-AUC	F1	PR-AUC	ROC-AUC	F1	PR-AUC
Deepwalk	0.8068	0.6474	0.7315	0.8043	0.6783	0.6521	0.6509	0.6615	0.7033
LINE	0.6726	0.4965	0.6142	0.7032	0.5932	0.5591	0.3154	0.461	0.4485
Node2vec	0.7346	0.5553	0.6427	0.8146	0.6831	**0.6818**	0.5822	0.6202	0.6362
GATNE	0.5878	0.485	0.4364	0.839	0.7	0.6789	0.8251	0.6857	0.6795
CTDNE	0.777	0.5565	0.6267	0.5351	0.4158	0.4125	0.6719	0.6791	0.7098
DynamicTriad	0.8111	0.598	0.6769	0.5717	0.4319	0.4612	0.6236	0.6366	0.6782
tNodeEmbed	0.7543	0.6737	0.7299	0.6519	0.5735	0.5182	**0.9495**	**0.8896**	0.9403
DynAERNN	0.7393	0.6367	0.6952	0.6861	0.5558	0.5481	0.8508	0.8266	0.8322
DyHINE	**0.8316**	**0.7533**	**0.7731**	**0.8715**	**0.7798**	**0.6818**	0.9013	0.8356	**0.9489**
DyHINE-e	0.8159	0.7245	0.7365	0.8648	0.7795	0.6793	0.8738	0.8194	0.9207
DyHINE-n	0.8249	0.7412	0.7398	0.8706	0.7793	0.6799	0.8654	0.8107	0.9143
%Improv.	2.53%	11.82%	5.17%	3.87%	11.4%	-	-	-	0.91%

according to the time series. After obtaining the representation of nodes, the inner product is used to perform temporal link prediction. Since this is a binary prediction problem, we use the area under the ROC curve (*ROC-AUC*), *F1* score, and PR curve (*PR-AUC*) as the evaluation methods. The Table 2 lists the results and the analysis are as follows:

- On Twitter and Amazon datasets, the performance of Deepwalk method is better than Node2vec, while on Movielens, the Node2vec method is superior to Deepwalk. It illustrates that Node2vec proposed biased random walk procedure to help improve the representation of nodes on the dense dataset and Deepwalk performs well with sparse data.
- LINE has a poor performance on Amazon. According to the feature analysis of datasets, we can find that it performs relatively well in dense data, but poorly in the case of sparse data and uneven distribution of edge types. It is more susceptible to data distribution than Deepwalk.
- GATNE performs well on relatively dense data. Compared with other methods mentioned above, it enhances learning by aggregating neighbor information, which to some extent reflects that aggregating neighbor node helps to improve embedding performance. However, it cannot learn good features in the especially sparse dataset, which may be because the introduction of other heterogeneous information in the case of insufficient data adds too much noise. Besides, GATNE combines with base embedding to integrate the relationship of different edges, which is better than the previous method, indicating the effectiveness of embedding different edges.
- On the whole, the dynamic method performs well for sparse data. By mining the data evolutionary features, this kind of method outperforms the static one by a substantial margin. Comparing among the four dynamic methods, The experiment results show that the tNodeEmbed method works better with more changing edges in a timing chip. Due to the rich variation in the time slice of the amazon data set, it has shown exceptionally excellent performance in the *ROC-AUC* and *F1* indexes. DynamicTriad and CTDNE perform well on sparse data. From the overall effect, tnodeEmbed's performance is better, to a certain extent reflects the effectiveness of the framework and the combination of history embedding can improve performance. This also brings us some inspiration when designing the model in combination with the previous node embedding aggregation.
- The experimental results show that our method performs better than the others as a whole and performs well in the case of very sparse data. This is because the hierarchical attention mechanism is used to strengthen the feature mining of the relationship between users. At the same time, taking into account the changes in the time dimension, our method has learned richer user features. Its performance on Amazon is not optimal, probably due to the loss of information in online updates.

In summary, DyHINE learns the better heterogeneous node embeddings than state-of-the-art methods on the temporal link prediction task, which lies in the better consideration of the network heterogeneity and dynamic challenge.

4.4 The Effect of Aggregating Edge Type and Neighbors (RQ2)

In order to judge the influence of edge aggregation and neighbor node aggregation on the model, we conduct ablation experiments to answer **RQ2**. DyHINE-e and DyHINE-n represent only using edge and node embedding for aggregation respectively. The experimental results are shown in Table 2.

- From the results, we find that the effect of aggregating neighbor node embedding is better than that of aggregating edge embedding on Twitter and Movielens dataset, but the results on Amazon are the opposite. This may be because the characteristics of neighbor nodes reflect user preferences, and when there are more edge types, they can effectively reflect better user preferences. In the actual situation, the use of attention mechanism to aggregate node information can reflect the importance of the connection relationships to a certain extent.
- It can be observed that using a single edge or node aggregation can also improve performance due to the use of side information.

Table 3: Average running time (seconds) per epoch under dynamic embedding model.

Methods	CTDNE	DynamicTriad	tNodeEmbed	DynAERNN	DyHINE
Twitter	2578	3	30	52	11
Movielens	11074	4	154	37	40
Amazon	10715	6	133	42	37

Many studies like GATNE [3] have also proved that using edge information can effectively improve model performance.

- It can be found that using hierarchical attention mechanism to aggregate both can effectively improve the performance. This may stem from using richer heterogeneous information and mining shallow relationships between nodes.

4.5 Efficiency Analysis (RQ3)

To show the computational efficiency of our proposed method to answer **RQ3**, the average time of all dynamic embedding methods run in GPU on different datasets are recorded in Table 3. The unit of time is minutes. It can be seen that our method can get relatively efficient performance. CTDNE takes longer time because it learns the evolution of continue-time, whereas the other baselines just learn a few snapshots. DynamicTriad seems to have excellent performance, but it cost lots of time during sampling triad. DyHINE costs longer time on Movielens due to the aggregation of neighbor nodes and edges and the dataset has more types of edge and node.

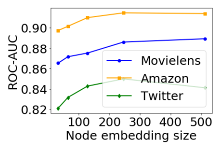
4.6 Parameter Sensitivity (RQ4)

In this part, the impact of embedding size on the model is explored by using the temporal link prediction task to answer **RQ4**. The experimental results are shown in Fig.4. The vertical comparison shows that the performance on Amazon dataset is better, while on Movielens is worse on *PR-AUC*. It can be observed that, as the embedding size increasing, the performance initially rises steadily, but then drops after the size reaching a certain level. Specifically, it can be concluded that the proposed model consistently achieves the best performance on three datasets when the node embedding size is between 200 and 300 and the edge embedding size is between 15 and 25. This desirable phenomenon may be contributed to the fact that the probability of assigning similar items to homogeneous embedding may decrease significantly when the embedding size is too large.

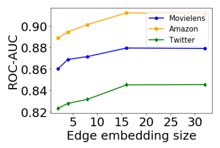
5 RELATED WORK

We review existing network embedding methods on homogeneous networks, heterogeneous networks and dynamic networks, which are most relevant to our work.

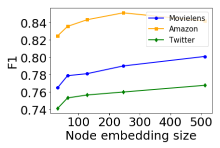
Homogeneous networks. Early work on homogeneous network embeddings rely on the skip-gram model to generate node representations, such as Deepwalk [18], Node2vec [10] and LINE [24]. NetMF [19] further theoretically unify these works into a matrix factorization framework. However, these methods can not generalize well to dynamic heterogeneous network since they ignore the heterogeneity and dynamic information of networks.



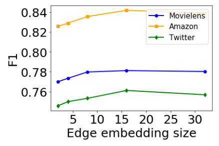
(a) Node: ROC-AUC



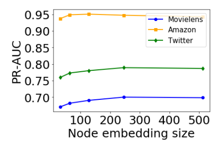
(b) Edge: ROC-AUC



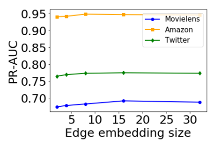
(c) Node: F1



(d) Edge: F1



(e) Node: PR-AUC



(f) Edge: PR-AUC

Figure 4: The impact of node and edge embedding size.

Heterogeneous networks. Recent studies [3–5, 8, 11, 14, 20, 27, 32, 35] have considered network embedding in heterogeneous networks. For example, PTE [23] represents markup information and word co-occurrence information at different levels as a large-scale heterogeneous text network and then embedded nodes into a low-dimensional space through a principled and efficient algorithm. Metapath2vec [5] introduces meta-path to learn node embeddings of heterogeneous information networks. MNE [32] represents each node by aggregating one common embedding and edge type embedding. HAN [27] uses hierarchical attention to aggregate information from the neighbors and combine various meta-paths. Despite the effectiveness of learning the heterogeneity of networks, these methods ignore the dynamic changes of networks.

Dynamic networks. Existing works [22, 25, 30, 38] focus on learning embeddings on dynamic homogeneous network. Multiple attempts have been devoted to embed time network, which can be roughly divided into two categories: embedded snapshot network [1, 6, 7, 15, 30, 36] and modeling temporal evolution [17, 22, 38]. For example, DynamicTriad [36] incorporates both structural information and evolution patterns of a given network based on the idea of triad. However, real-world networks, such as social network of friends and e-commerce network, keeps changing over time and few work [1] have study the dynamic embedding methods on heterogeneous network. The existing work [1] uses meta-path to learn about the heterogeneity of the network based on the DynamicTriad [36].

Above methods either learn node embedding on the snapshot or model the network only in continuous time. Most studies do not consider how to efficiently update embeddings when the number of nodes increases. Our method solves the problem of efficiently updating embedding while considering the time dimension.

6 CONCLUSION

In this paper, we generalize the problem of learning and updating embedding of dynamic heterogeneous networks and then propose DyHINE with the dynamic time-series embedding and online real-time update model. The target node embedding is divided into two parts: its own features and aggregated features. The aggregated features consist of neighbor node and edge embedding, and the hierarchical attention mechanism is adopted to aggregate them. In the learning of temporal information, We use temporal random walk to learn them better. The operator is designed in this paper to solve the frequently changing problem of dynamic network. Only some nodes were dynamically modified to avoid rerunning the whole network. In this paper, three real-world datasets are used in the experiments and the results show that DyHINE method performs better on the temporal link prediction task than existing methods to some extent. In addition, some characteristics of other models are found.

This work explores the potential of dynamic network embedding learning and updating. There are many directions that can be improved in the future. At first, the validity of the LINE model can be considered to extend DyHINE to the model with specified neighbors' order. Secondly, we can focus on the sparse problem and introduce cross-domain data to assist embedding generation. Thirdly, we can design better strategies to identify affected nodes to improve the performance of online real-time update model.

ACKNOWLEDGMENTS

We thanks the anonymous reviewers for their useful suggestions. The paper was supported by the Key-Area Research and Development Program of Guangdong Province (2020B010165003), the National Natural Science Foundation of China (61702568, U1711267), the Guangdong Basic and Applied Basic Research Foundation (2020A 1515010831), the Program for Guangdong Introducing Innovative and Entrepreneurial Teams (2017ZT07X355), and the Key-Area Research and Development Program of Guangdong Province of China (2018B030325001).

REFERENCES

[1] Ranran Bian, Yun Sing Koh, Gillian Dobbie, and Anna Divoli. 2019. 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*. ACM, ACM, 861–864.

[2] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2016. Deep neural networks for learning graph representations. In *AAAI*. AAAI Press.

[3] Yukuo Cen, Xu Zou, Jianwei Zhang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Representation Learning for Attributed Multiplex Heterogeneous Network. *arXiv preprint arXiv:1905.01669* (2019).

[4] Hongxu Chen, Hongzhi Yin, Weiqing Wang, Hao Wang, Quoc Viet Hung Nguyen, and Xue Li. 2018. PME: projected metric embedding on heterogeneous networks for link prediction. In *KDD*. ACM, 1177–1186.

[5] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *KDD*. ACM, 135–144.

[6] Lun Du, Yun Wang, Guojie Song, Zhicong Lu, and Junshan Wang. 2018. Dynamic Network Embedding: An Extended Approach for Skip-gram based Network Embedding.. In *IJCAI*. 2086–2092.

[7] Nan Du, Yichen Wang, Niao He, Jimeng Sun, and Le Song. 2015. Time-sensitive recommendation from recurrent user activities. In *Advances in Neural Information Processing Systems*. 3492–3500.

[8] Xinyu Fu, Jiani Zhang, Ziqiao Meng, and Irwin King. 2020. MAGNN: Metapath Aggregated Graph Neural Network for Heterogeneous Graph Embedding. In *Proceedings of The Web Conference 2020*. 2331–2341.

[9] Palash Goyal, Sujit Rokka Chhetri, and Arquimedes Canedo. 2020. dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. *Knowledge-Based Systems* 187 (2020), 104816.

[10] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *KDD*. ACM, 855–864.

[11] Binbin Hu, Chuan Shi, Wayne Xin Zhao, and Philip S Yu. 2018. Leveraging meta-path based context for top-n recommendation with a neural co-attention model. In *KDD*. ACM, 1531–1540.

[12] Srijan Kumar, Xikun Zhang, and Jure Leskovec. 2019. Predicting dynamic embedding trajectory in temporal interaction networks. In *KDD*. ACM, 1269–1278.

[13] Taisong Li, Jiawei Zhang, S Yu Philip, Yan Zhang, and Yonghong Yan. 2018. Deep dynamic network embedding for link prediction. *IEEE Access* 6 (2018), 29219–29230.

[14] Weiyi Liu, Pin-Yu Chen, Sailung Yeung, Toyotaro Suzumura, and Lingli Chen. 2017. Principled multilayer network embedding. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, 134–141.

[15] Jianxin Ma, Peng Cui, and Wenwu Zhu. 2018. Depthlgp: Learning embeddings of out-of-sample nodes in dynamic networks. In *AAAI*.

[16] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[17] Giang Hoang Nguyen, John Boaz Lee, Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, and Sungchul Kim. 2018. Continuous-time dynamic network embeddings. In *Companion Proceedings of the The Web Conference 2018*. International World Wide Web Conferences Steering Committee, 969–976.

[18] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 701–710.

[19] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. 2018. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 459–467.

[20] Meng Qu, Jian Tang, Jingbo Shang, Xiang Ren, Ming Zhang, and Jiawei Han. 2017. An attention-based collaboration framework for multi-view network representation learning. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 1767–1776.

[21] Chuan Shi, Binbin Hu, Wayne Xin Zhao, and S Yu Philip. 2018. Heterogeneous information network embedding for recommendation. *IEEE Transactions on Knowledge and Data Engineering* 31, 2 (2018), 357–370.

[22] Uriel Singer, Ido Guy, and Kira Radinsky. 2019. Node Embedding over Temporal Graphs. (2019), 4605–4612.

[23] Jian Tang, Meng Qu, and Qiaozhu Mei. 2015. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *KDD*. ACM, 1165–1174.

[24] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*. International World Wide Web Conferences Steering Committee, 1067–1077.

[25] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. 2019. DyRep: Learning Representations over Dynamic Graphs. (2019).

[26] Hongwei Wang, Fuzheng Zhang, Min Hou, Xing Xie, Minyi Guo, and Qi Liu. 2018. Shine: Signed heterogeneous information network embedding for sentiment link prediction. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 592–600.

[27] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous Graph Attention Network. In *The World Wide Web Conference*. ACM, 2022–2032.

[28] Carl Yang, Mengxiong Liu, Zongyi Wang, Liyuan Liu, and Jiawei Han. 2017. Graph clustering with dynamic embedding. *arXiv preprint arXiv:1712.08249* (2017).

[29] Jiaxuan You, Yichen Wang, Aditya Pal, Pong Eksombatchai, Chuck Rosenburg, and Jure Leskovec. 2019. Hierarchical Temporal Convolutional Networks for Dynamic Recommender Systems. In *The World Wide Web Conference*. ACM, 2236–2246.

[30] Wenchao Yu, Wei Cheng, Charu C Aggarwal, Kai Zhang, Haifeng Chen, and Wei Wang. 2018. Netwalk: A flexible deep embedding approach for anomaly detection in dynamic networks. In *KDD*. ACM, 2672–2681.

[31] Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. 2018. Network representation learning: A survey. *IEEE transactions on Big Data* (2018).

[32] Hongming Zhang, Liwei Qiu, Lingling Yi, and Yangqiu Song. 2018. Scalable Multiplex Network Embedding.. In *IJCAI*, Vol. 18. 3082–3088.

[33] Yizhou Zhang, Yun Xiong, Xiangnan Kong, Shanshan Li, Jinhong Mi, and Yangyong Zhu. 2018. Deep collective classification in heterogeneous information networks. In *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee, 399–408.

[34] Ziwei Zhang, Peng Cui, Jian Pei, Xiao Wang, and Wenwu Zhu. 2018. Timers: Error-bounded svd restart on dynamic networks. In *AAAI*. AAAI Press.

[35] Kai Zhao, Ting Bai, Bin Wu, Bai Wang, Youjie Zhang, Yuanyu Yang, and Jian-Yun Nie. 2020. Deep Adversarial Completion for Sparse Heterogeneous Information Network Embedding. In *Proceedings of The Web Conference 2020*. 508–518.

[36] Lekui Zhou, Yang Yang, Xiang Ren, Fei Wu, and Yueting Zhuang. 2018. Dynamic network embedding by modeling triadic closure process. In *AAAI*.

[37] Dingyuan Zhu, Peng Cui, Ziwei Zhang, Jian Pei, and Wenwu Zhu. 2018. High-order proximity preserved embedding for dynamic networks. *IEEE Transactions on Knowledge and Data Engineering* 30, 11 (2018), 2134–2144.

[38] Yuan Zuo, Guannan Liu, Hao Lin, Jia Guo, Xiaoqian Hu, and Junjie Wu. 2018. Embedding temporal network via neighborhood formation. In *KDD*. ACM, 2857–2866.