Expert Systems With Applications 158 (2020) 113427

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

# HIN_DRL: A random walk based dynamic network representation learning method for heterogeneous information networks

LU Meilian*, YE Danna

*State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China*

ARTICLE INFO

*Article history:*
Received 13 June 2019
Revised 30 March 2020
Accepted 31 March 2020

*Keywords:*
Dynamic representation learning
Heterogeneous information networks
Meta path
Dynamic random walk

ABSTRACT

Learning the low-dimensional vector representation of networks can effectively reduce the complexity of various network analysis tasks, such as link prediction, clustering and classification. However, most of the existing network representation learning (NRL) methods are aimed at homogeneous or static networks, while the real-world networks are usually heterogeneous and tend to change dynamically over time, therefore providing an intelligent insight into the evolution of heterogeneous networks is more practical and significant. Based on this consideration, we focus on the dynamic representation learning problem for heterogeneous information networks, and propose a random walk based Dynamic Representation Learning method for Heterogeneous Information Networks (HIN_DRL), which can learn the representation of network nodes at different timestamps. Specifically, we improve the first step of the existing random walk based NRL methods, which generally include two steps: constructing node sequences through random walk process, and then learning node representations by throwing the node sequences into a homogeneous or heterogeneous Skip-Gram model. In order to construct optimized node sequences for evolving heterogeneous networks, we propose a method for automatically extracting and extending meta-paths, and propose a new method for generating node sequences via dynamic random walk based on meta-path and timestamp information of networks. We also propose two strategies for adjusting the quantity and length of node sequences during each random walk process, which makes it more effective to construct the node sequences for heterogeneous information networks at a specific timestamp, thus improving the effect of dynamic representation learning. Extensive experimental results show that compared with the state-of-art algorithms, HIN_DRL achieves better results in Macro-F1, Micro-F1 and NMI for multi-label node classification, multi-class node classification and node clustering on several real-world network datasets. Furthermore, case studies of visualization and dynamic on Microsoft Academic dataset demonstrate that HIN_DRL can learn network representation dynamically and more effectively.



## 1. Introduction

With the rapid development of information networks, a major challenge for researchers is how to improve the efficiency of various machine learning tasks, such as node similarity measure, link prediction, classification and clustering, in large-scale information networks. Traditional graph-based representation methods typically rely on the adjacency matrix, adjacency table, or carefully designed features (Goyal & Ferrara, 2018). However, manually designed features are inflexibility and designing them may be time-consuming and expensive, causing the limitations to analyze networks in depth.

In response to above problem, a variety of recent studies have focused on network representing learning (NRL) which can learn the low-dimensional vector representation of nodes or edges in information networks and thus effectively reduce the complexity of various machine learning tasks based on networks. That is, to learn a mapping, where nodes or sub-graphs or even entire graphs are represented as points in a low-dimensional space by preserving the structure of original networks and the information they contain as much as possible. Then these learned low-dimensional vectors can be directly considered as the network features for downstream machine learning tasks, such as link prediction, classification, clustering, etc.

Most of the network representation learning (NRL) methods that have been proposed so far are for static homogeneous information networks, such as LLE (Roweis & Saul, 2000), DeepWalk (Perozzi, Al-Rfou & Skiena, 2014), and LINE (Tang et al., 2015a),

* Corresponding author.
*E-mail addresses:* mllu@bupt.edu.cn (L. Meilian), yedanna@bupt.edu.cn (Y. Danna).

https://doi.org/10.1016/j.eswa.2020.113427
0957-4174/

etc. However, the real-world networks are usually heterogeneous and could change dynamically. These methods can effectively represent network structure, but they could lose some important information for heterogeneous networks.

As a powerful representation of complex networks and multi-type data, heterogeneous information networks have been widely studied in recent years. The NRL methods for heterogeneous information networks include HINE (Huang & Mamoulis, 2017), Metapath2vec (Dong, Chawla & Swami, 2017), etc., which extract rich network semantic information by utilizing meta-paths, thus achieving a better network representation.

However, the rich semantic information brought by multi-type nodes and multi-type relationships in heterogeneous information networks has seriously increased the difficulty of NRL problem, and most existing NRL methods for heterogeneous information networks also have some limitations, such as only can be applied to a specific type of network, only utilize network structure and meta-paths, while neglect dynamic characteristics of networks. Meanwhile, some existing solutions, such as HINE (Huang & Mamoulis, 2017) and Metapath2vec (Dong, Chawla & Swami, 2017), fail to consider that the node representation may change due to the evolution of networks.

Although some solutions can solve dynamic NRL problem, such as GraphSAGE (Hamilton, Ying, & Leskovec, 2017), most of them achieve that by considering dynamic networks as a sequence of discrete snapshot graphs and then predict the network representation in the future, which would greatly increase the learning cost. In addition, most of the dynamic NRL methods such as DepthLGP (Ma, Cui & Zhu, 2018) are based on dynamic homogeneous information networks, and some methods for heterogeneous information networks only involve two types of nodes, such as DeepCoevolve (Dai, Wang, Trivedi, & Song, 2017), resulting constrained application scenarios.

In order to be closer to the real-world network scenarios, we focus on the problem of dynamic NRL for evolving heterogeneous information networks instead of static networks or homogeneous networks. The work most related to ours is Metapath2vec (Dong, Chawla & Swami, 2017) proposed by Dong et al. They first generate node sequences with network semantics for various types of nodes by utilized meta-paths to guide random walk process, and then extends the Skip-Gram model to solve the network heterogeneity, thus improving the effect of representation learning. However, the meta-path based random walk strategy proposed by Dong et al. does not consider the influence of time information on representation learning, nor does it consider how to avoid generating the possible noise node pairs during static random walk process, while these noise node pairs would reduce the performance of node representation learned by Skip-Gram model.

To overcome the above limitations in Metapath2vec, we propose a new random walk based dynamic representation learning method for heterogeneous information networks, HIN_DRL. It can learn the dynamic representation of nodes for different types of heterogeneous information networks at different timestamps, thereby facilitating subsequent machine learning tasks in evolving heterogeneous information networks. HIN_DRL improves the first phase of the existing random walk based NRL methods that generally include two steps: the first step is constructing node sequences through random walk process, and the second step is learning node representations by throwing the node sequences into a homogeneous or heterogeneous Skip-Gram model. Specifically, for constructing optimized node sequences, we first propose an improved method for searching and extending meta-paths with any types of head and tail nodes based on the method of auto-mining meta-paths with the same types of head and tail nodes proposed in HeteClass (Gupta, Kumar & Bhasker, 2017), thus ensuring that the extracted meta-paths can cover richer and more objective semantic information. Then, a method guided by meta-paths, called Edge-based Dynamic Random Walk (EDRW), is designed for generating node sequences. EDRW calculates the similarity between nodes by integrating topic vectors of nodes and timestamps of edges, calculates edge-to-edge transition probabilities by combining the similarity between any two nodes within two hops, and dynamically adjusts the quantity and length of node sequences utilizing two edge timestamps based random walk control strategies proposed in this paper, thus generating optimized node sequences at different timestamps. Unlike Metapath2vec (Dong, Chawla & Swami, 2017), EDRW guides edge-based random walk instead of node-based random walk along meta-paths, which guarantees that the generated node sequences can simultaneously contain richer network structure and semantic information, thus improving the effect of NRL.

Our contributions can be summarized as following:

(1) We propose an improved automatic meta-path searching method (AutoMP) and corresponding extension strategies for all three different types of initial meta-paths in heterogeneous information networks, thus ensuring that the extracted meta-paths can cover richer and more objective semantic information, thus the subsequent random walk can be conducted in any types of heterogeneous information networks.
(2) An edge-to-edge similarity measurement method based on network structure, timestamps and text information, as well as an edge-based dynamic heterogeneous random walk method for generating node sequences under the guidance of extended meta-paths are proposed, thus the generated node sequence can carry rich structural and semantic information.
(3) Two edge timestamps based random walk optimization strategies, namely LOS and QOS respectively, are proposed to further control the random walk process, thus optimizing the quantity and length of the generated node sequences dynamically.
(4) The effects of HIN_DRL on multiple machine learning tasks including multi-label node classification, multi-class node classification, node clustering and visualization are conducted on several real-world network datasets. The experimental results show that compared with the state-of-art methods, HIN_DRL can learn higher quality low-dimensional representation of network nodes and thus effectively improving the effect of subsequent machine learning tasks. Besides, HIN_DRL can learn different node representation over time, thus can be applied to analyze the dynamics of heterogeneous information networks and discovering the community or behavior characteristics via visualization technology.

The remainder of this paper is organized as follows. Section 2 reviews the related work. Section 3 gives the related definitions and problem description. Our proposed HIN_DRL is elaborated in Section 4. The experimental results are reported in Section 5. Finally, Section 6 concludes the paper and gives future work.

## 2. Related work

### 2.1. Homogeneous network representation learning

Early NRL work mainly focused on homogeneous information networks, and the main challenge is how to preserve the network structure information as much as possible.

Matrix factorization-based NRL methods generally construct a relation matrix first, and then decompose the relation matrix to obtain the low-dimensional representation of network nodes. These methods mainly concentrate on improving the construction of relation matrix, such as LE (Anderson & Morley, 1985), LLE (Roweis & Saul, 2000), and GraRep (Cao, Lu & Xu, 2015), etc. Differently, NEU (Yang et al., 2017) first obtains the low dimensions

of a low-order approximation matrix, and then makes the low-order matrix approximate the result of a high-order approximation through an update operation $R' = R + \lambda A \bullet R$. These matrix factorization-based methods consider the global similarity of network nodes, but their time complexity and space complexity are too high to be applied to large-scale networks.

Random walk-based NRL methods generally obtain a large number of node sequences first, and then obtain the low-dimensional representation of network nodes by throwing these sequences into the Skip-Gram model of Word2vec (Mikolov et al., 2013). Perozzi et al. proposed DeepWalk (Perozzi, Al-Rfou & Skiena, 2014), which learns the low-dimensional representation of network nodes utilizing Word2vec model by analogically considering network nodes as words and the node sequences generated by random walk process as the sentences of documents. Some other methods were subsequently proposed, such as Node2vec (Grover & Leskovec, 2016), which extracts network structure information by combining the strategies of BFS and DFS; Struc2vec (Ribeiro et al., 2017), which can identify node pairs that are far apart in the network but play similar roles; Walklets (Perozzi, Kulkarni & Skiena, 2016), which constructs different sampling space by generating multiple sets of different node sequences with $k$ hops each step; and CARE (Keikha, Rahgozar & Asadpour, 2018), which considers both edge and community information to preserve local and global network structures by defining different walk probability. Since these random walk-based methods always consider local network information, it is difficult to design an optimal node sequences sampling strategy for them. However, their time complexity are relatively low, and thus can be applied to large-scale networks.

The outstanding achievements of neural networks in the fields of natural language processing and image recognition promote the application of deep learning models in NRL. SDNE (Wang, Cui & Zhu, 2016) models second-order proximity to capture global network structure information by utilizing unsupervised deep auto-encoders and models first-order proximity to preserve local network structure information by utilizing a supervised Laplace matrix. SENN (Qiao et al., 2019) first captures different order relationships between network nodes, then adopts a stacked denoising auto-encoder to reduce the noise data and learn the low-dimensional representation of network nodes, finally an attention mechanism is further used to combine different node representation and thus obtain the final node representation. Although these deep learning-based methods can learn more complex and accurate network representation, their costs are expensive. For example, a large number of parameters need to be learnt, so their time complexity is very high and expensive hardware support such as GPU is generally required.

In addition, there are other ways to deal with the representation learning problem for homogeneous information networks. For example, LINE (Tang et al., 2015a) models the relationships between nodes by considering first-order proximity and second-order proximity, which has low time complexity and can be well applied to large-scale networks. RUM (Yu et al., 2017) further takes community affiliation into account, thus retains the network structure information more comprehensively. However, most of these methods only consider network structure information while neglect a variety of other available and valuable information such as text information and labels.

### 2.2. Heterogeneous network representation learning

The challenges of NRL for heterogeneous information networks mainly come from how to deal with and balance the relationships between different types of nodes or edges.

TransNet (Tu et al., 2017) was proposed to predict labels for unlabeled edges in social networks, which first auto-encodes the labeled edges to obtain their low-dimensional representation, then gets the vector representation of the nodes that make up the edges by utilizing transformation mechanism. Finally, the unlabeled edges can be represented by the vector representation of nodes, and their original label sets can be restored by utilizing a decoder. Tri-DNR (Pan et al., 2016) learns node representation by combining the network structure information, text information and label information of nodes and captures node-node, node-word, and label-word relationships by coupling two shallow neural networks. These deep learning-based methods can effectively mine the nonlinear relationships contained in networks, but they need to learn a lot of parameters, which would lead to high computational cost.

Tang et al. proposed PTE (Tang et al., 2015b) by migrating LINE (Tang et al., 2015a) from homogeneous networks to heterogeneous networks, which first constructs three different bipartite graphs using text corpus, then learns the low-dimensional representation of words, documents and labels by applying LINE to these graphs independently. Xu et al. proposed EOE (Xu et al., 2017a) for coupled heterogeneous networks, where inter-network edges can provide supplemental information to intra-network edges, thereby solving the cold-start problem. Moreover, for the problem of unexplainable implicit representations, Xu et al. pointed out that analyzing an action should consider two aspects (Xu et al., 2017b), the characteristics of who, such as education, family, etc., and the characteristics of what, such as attitude, interests, and so on. Besides, Huang et al. proposed HINE (Huang & Mamoulis, 2017), an effective heterogeneous network embedding method based on the proximity measure of meta-paths. These methods are generally based on a proximity relationship, such as first-order proximity or second-order proximity, and obtain the low-dimensional representation by minimizing the distance between their theoretical and empirical values; or obtain auxiliary information such as node attributes and label information by combining multi-network fusion technology, and thus improving the embedding effect.

Similar to the random walk-based methods for homogeneous information networks, Word2vec model can also be used to perform NRL for heterogeneous information networks. Dong et al. (Dong, Chawla & Swami, 2017) first proposed a meta-path based random walk method to generate node sequences, and designed a heterogeneous Skip-Gram model and a new heterogeneous negative sampling method for the node sequences with multiple node types, thus extending the original Skip-Gram model to heterogeneous information networks. Pham et al. (Phu Pham & Phuc Do, 2019) recently proposed a topic-driven meta-path based model, W-MetaPath2Vec, which enhances the representation learning of heterogeneous information networks by combining the topic similarity between nodes with semantic correlations. Besides, Chen et al. proposed WTL+IBL (Chen et al., 2017) to conduct NRL for e-commerce networks, where WTL generates node sequences via weighted random walk method, and IBL distinguishes different types of nodes by considering that different types of nodes carry different attributes.

Qu et al. (2017) proposed the concept of multi-view node similarity and measured the weight of multiple similarity by utilizing attention mechanism, thus mining various relationships between nodes. Qu et al. further suggested that the study can be extended to heterogeneous information networks by utilizing the diversity of meta-paths to guide different similarity measures. In addition, Ma et al. proposed MVGE-HD (Ma et al., 2017), which for the first time considers the hub detection mechanism when solving multi-view graph embedding. Gui et al. (2017) pointed out that, although meta-paths can represent different relationships, they may lead to the loss of some subtle information. In order to solve such problems, Gui et al. represented heterogeneous events using hyper-

graphs and hyper-edges. However, these methods fail to consider the dynamics of networks.

### 2.3. Dynamic network representation learning

The main challenge of NRL for evolving networks comes from how to take time information in networks into account, thus to capture the evolution of networks and learn the low-dimensional representation of the nodes that have changed.

Ma et al. proposed DepthLGP (Ma, Cui & Zhu, 2018), which first learns the potential node representation via Laplace Gaussian process that can preserve the important node properties, then converts the potential representation into a low-dimensional representation by utilizing a deep neural network. Yu et al. (2018) proposed the method NetWalk to detect abnormal nodes in dynamic networks by learning the representation of networks. They first proposed a new dimensionality reduction algorithm, clique embedding, to learn the low-dimensional representation of nodes based on a deep auto-encoder neural network by minimizing the pairwise distance between all nodes in each node sequence. Then, a clustering-based technique is employed to dynamically detect network anomalies.

Nguyen et al. (2018) considered that a temporal random walk can reduce the uncertainty of walkers and proposed a general NRL method for continuous-time dynamic networks, CTDNE.

Zhu et al. (2018) proposed an efficient NRL method for dynamic homogeneous information networks, DHPE. The method first adopts generalized SVD (GSVD) to preserve the high-order proximity. Then, by transforming the GSVD problem into a generalized eigenvalue problem and utilizing the matrix perturbation theory, the results of GSVD are updated to incorporate the network changes. Besides, Zhu et al. proposed an accelerated solution to improve the efficiency of DHPE.

GraphSAGE proposed by Hamilton et al. (2017) generates low-dimensional representation by sampling and aggregating features from a node's local neighbors. The goal of this method is to learn a set of model parameters rather than the low-dimensional representation for each node individually. Therefore, when a new node appears, its low-dimensional representation can be obtained directly without retraining the model. Du et al. (2018) proposed an extended dynamic network embedding framework for Skip-gram based methods, DNE. They first introduced a decomposable objective equivalent to the objective of LINE (Tang et al., 2015a) to learn the representation of network nodes, and then learn the representation of new nodes based on the objective. They further chose the greatly affected nodes between two snapshots to update, then treat these nodes as new nodes and update their representations.

Zhou et al. (2018) proposed the DynamicTriad, which models the evolution patterns of networks using triad (i.e., group of three nodes). They believed that the triadic closure process is a fundamental mechanism during the formation and evolution of networks, so DynamicTriad can capture the dynamics of networks and learn the node representations at different time.

However, the above methods cannot deal with the heterogeneity of information networks, thus fail to be directly applied to heterogeneous information networks.

Considering that different types of nodes carry different attributes and the attributes may change over time, Li et al. proposed a novel dynamic attributed network embedding framework, DANE (Li et al., 2017), which for the first time provides an offline NRL method to preserve node proximity, such as network structures and node attributes, and then updates network embedding online by utilizing matrix perturbation theory based on the changes in networks. Dai et al. proposed DeepCoevolve (Dai et al., 2017), which performs representation learning for both users and products based on their interaction graph. DeepCoevolve defines the intensity function in point processes over evolving networks by utilizing recurrent neural network (RNN), which can capture complex mutual influence between users and products, as well as the evolution of their features over time. Y. Zuo et al. introduced the concept of neighborhood formation sequence to describe the evolution of a node, where there are temporal excitation effects between neighbors. They further proposed HTNE (Zuo et al., 2018), which learns the node embedding by modeling the neighborhood formation sequence as a Hawkes process and captures the influence of historical neighbors on the current neighbors simultaneously.

However, these methods for dynamic heterogeneous information networks have some shortcomings: DANE has large time complexity because it involves matrix calculation; DeepCoevolve learns the representation of specific new users and new products conditionally, that is, only when new users or new products interact with an existing product or user, it can trigger an update of the representation; HTNE is only applicable to homogeneous information networks or heterogeneous information networks with only two types of nodes.

Considering that the random walk based methods have low time complexity, we propose a random walk based dynamic network representation learning method for heterogeneous information networks, HIN_DRL, which can learn node representation for evolving heterogeneous information networks at different timestamps. Specifically, HIN_DRL generates optimized node sequences by utilizing edge-to-edge based dynamic random walk guided by meta-paths, so the generated node sequences could reflect the heterogeneity and dynamics of networks, and thus can be used to learn the representation of different types of nodes at different timestamps.

## 3. Terms definition

**Definition 3.1 Heterogeneous Information Network** is defined as a graph $G = (V, E, A, R)$, where each node $v \in V$ denotes an entity, each edge $e \in E(e = (v, v'))$ denotes a relation, each $A_i \in A$ denotes an entity type and each $R_i \in R$ denotes a relation type. There is a mapping function associated with each node and each edge respectively, namely $\Phi: V \to A$ and $\Psi: E \to R$, in which $|A| > 1$ or $|R| > 1$. (Sun et al., 2009)

**Definition 3.2 Dynamic Heterogeneous Information Network** is defined as a graph $G_T = (V, E, A, R, T)$, in which $T$ denotes the timestamp set and each $t \in T$ is a timestamp existing in $G_T$. There is a mapping function $\Upsilon: E \to T$, and $\forall e \in E$, $\Upsilon(e) \in T$ denotes each edge in the network is associated with a timestamp.

**Definition 3.3 Dynamic Network Schema** is defined as $T_G = (A, R, T)$, which denotes the meta information of network $G_T = (V, E, A, R, T)$, including entity type $A$, relation type $R$ and timestamp $T$.

**Definition 3.4 Center Type Node Edge Type Node** andone kind of special nodes called *Cnode* which carry text information and timestamp in $T_G = (A, R, T)$. In this paper, we focus on the NRL problem for heterogeneous information networks containing center type nodes. For example, center type nodes are those paper nodes in heterogeneous academic networks, those blog nodes in traditional social networks, and those review nodes in LBSNs. The remaining nodes in the network are **Edge Type Node**, called *Enode*.

**Definition 3.5 Hub Node** is the node which links two edges. For example, if there are two edges $e(a, b)$ and $e'(b, c)$ in a network, then node $b$ is the hub node of nodes $a$ and $c$. Hub node provides reachable paths for their neighbor nodes in edge-based random walk.

**Definition 3.6 Meta-path** is a sequence of relationships consisting of a series of entity types that characterize heterogeneous information networks. Given a network schema $T_G = (A, R)$, a meta-path can be formally described as $A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2}$

$\cdots \xrightarrow{R_l} A_{l+1}$. Where $R = R_1 \circ R_2 \circ \cdots R_l$ denotes a composite relation between entity types $A_1$ and $A_{l+1}$, $\circ$ denotes the composite operator (Sun et al., 2011). Meta-paths can be distinguished as symmetric meta-paths and asymmetric meta-paths based on the difference of the constituent elements, in which **symmetric meta-path** consists of symmetric relation types and can be denoted as $A_1 \xrightarrow{R_1} \cdots \xrightarrow{R_{k-1}} A_k \xrightarrow{R_k} A_{k+1} \xrightarrow{R_k^{-1}} A_k \xrightarrow{R_{k-1}^{-1}} \cdots \xrightarrow{R_1^{-1}} A_1$. Correspondingly, **asymmetric meta-path** consists of asymmetric relation types, which is further distinguished as asymmetric meta-paths with the same types of head and tail entities, such as meta-path *PPVP* in heterogeneous academic networks, and asymmetric meta-paths with different types of head and tail entities, like meta-path *APV* in heterogeneous academic network. These two kinds of asymmetric meta-paths are respectively denoted as $A_1 \xrightarrow{R_1} \cdots \xrightarrow{R_{(t-1)}} A_t \xrightarrow{R_t} A_{t+1} \xrightarrow{R_k} A_k \xrightarrow{R_{(k-1)}} \cdots \xrightarrow{R_s} A_1$ and $A_1 \xrightarrow{R_1} \cdots \xrightarrow{R_{(k-1)}} A_k \xrightarrow{R_k} A_{k+1}$.

**Definition 3.7 Dynamic Representation Learning for Heterogeneous Information Network.** Given a dynamic heterogeneous information network $G_T$ and a specified timestamp $t$, a low-dimensional latent mapping function $f_G^t : V \to X, X \in \mathbb{R}^{|V| \times d}, d << |V|$ can be learned to represent each node in $G_T$ with a $d$ dimensional eigenvector at timestamp $t$.

## 4. Proposed method HIN_DRL

In this paper, we propose a dynamic NRL method for heterogeneous information networks, HIN_DRL, which includes two stages: dynamically generating node sequences and learning the low-dimensional representation of network nodes. Specifically, we first propose an edge-based dynamic random walk method guided by meta-paths, EDRW, to generate optimized node sequences, and the edge-based transition probability of random walk is measured by utilizing network structures, timestamps and text information. In addition, two random walk control strategies based on timestamps, called LOS and QOS, are proposed to adjust the length and number of node sequences in real time respectively, thus the node sequences of dynamic information networks can be effectively constructed at different timestamps. Then the generated node sequences are thrown into a heterogeneous Skip-Gram model to learn the low-dimensional vectors of network nodes.

The framework of HIN_DRL is shown in Fig. 1, which consists of four steps:

(1) Constructing a dynamic heterogeneous information network by specifying a corresponding network schema for a dataset and extracting information such as entities, relations, and timestamps according to the network schema.
(2) Based on the specified network schema, automatically searching initial symmetric and asymmetric meta-paths by utilizing AutoMP which is an improved meta-path searching method proposed in this paper.
(3) Generating node sequences, including constructing an edge-based transition probability matrix for dynamic random walk and generating node sequences via meta-path guided dynamic random walk.
(4) Learning the low-dimensional vectors of nodes in dynamic heterogeneous information networks by throwing the generated node sequences into a heterogeneous Skip-Gram model.

### 4.1. Constructing a dynamic heterogeneous information network

We first specify a dynamic network schema $T_G = (A, R, T)$ based on the source dataset and determine the entity types and relation types. Then, according to the network schema, we extract nodes and edges and their attributes from the dataset to construct a dynamic heterogeneous information network. In which, the content attributes, such as the abstracts and keywords of paper nodes in academic dataset, are used to generate topic vectors of nodes through a topic model, and the topic vectors are then used for measuring the similarity between nodes. The timestamp attributes, such as the publication time of papers in academic dataset, are used to calculate the similarity between nodes and that between edges on the one hand, and to control the edge-based random walk process for generating node sequences on the other hand.

### 4.2. Searching initial meta-paths

Inspired by Metapath2vec (Dong, Chawla & Swami, 2017), we use meta-paths to guide the random walk process, thus to construct node sequences more efficiently. Considering that manually

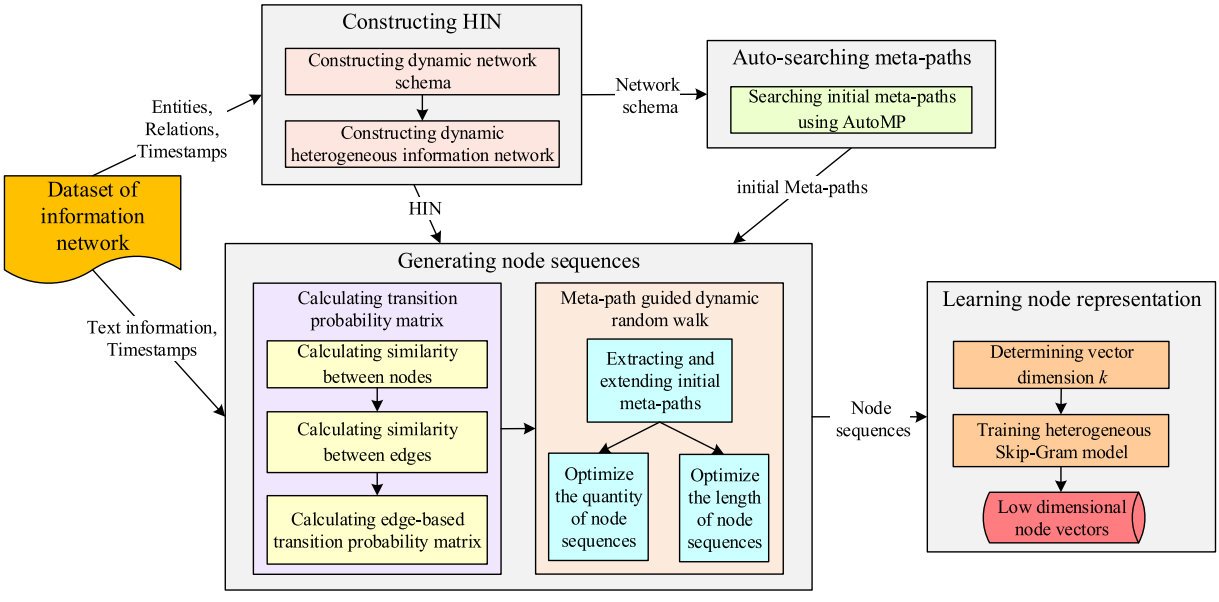
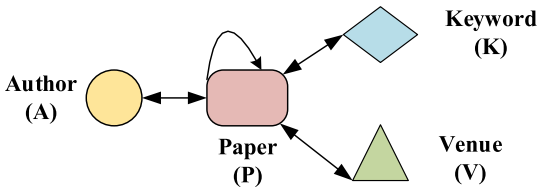
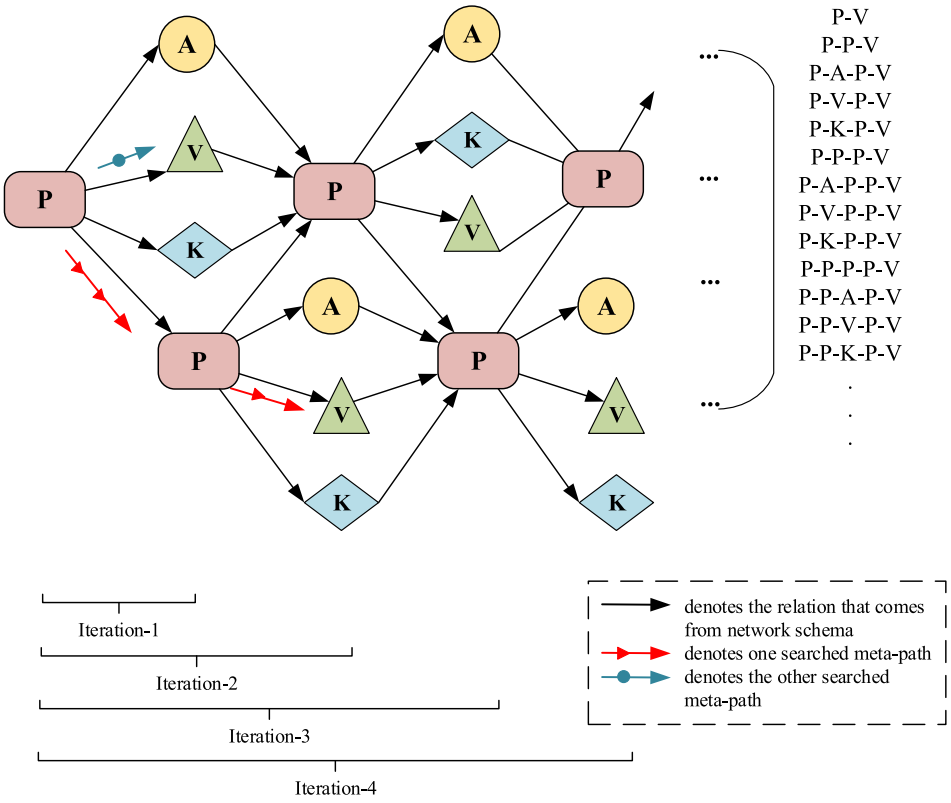



**Fig. 1.** Framework of HIN_DRL.




(a) Network schema




(b) Searching meta-paths

**Fig. 2.** Example of automatically searching initial meta-paths using AutoMP.

specifying meta-paths is usually insufficient, and the meta-path searching method proposed by Gupta, Kumar & Bhasker (2017) can only extract meta-paths with the same type of head and tail nodes rather than those with different types of head and tail nodes, we propose an improved method, AutoMP, for automatically extracting any types of meta-paths, including symmetric meta-paths and asymmetric meta-paths with the same or different types of head and tail nodes, thus improving the efficiency and quality of semantic searching in heterogeneous information networks. Furthermore, to minimize the space-time complexity of extracting meta-paths and the subsequent random walk process based on meth-paths, AutoMP only searches the meta-paths within a limited length.

Fig. 2 shows the procedure that AutoMP searches meta-paths, where the black lines indicate the relations that come from the network schema, the red lines marked with triangles denote the searched meta-paths and the blue lines marked with circles denote the other searched meta-paths. Suppose that a heterogeneous academic network schema is given in Fig. 2(a), the maximum length of the meta-paths to be searched is 4, and the head node type and tail node type are P and V respectively, then AutoMP will perform 4 iterations.

In the first iteration, the one-hop meta-paths with head node type P is extracted, which are denoted as firstedge_set={PA, PV, PK, PP}. Then, the meta-paths with head node type P and tail node type V in firstedge_set are extracted, which are denoted as IniPathSet={PV}.

In the second iteration, the tail node types of all elements in firstedge_set are extracted and denoted as targetNodes={A, V, K, P}, which are then considered as the head node types for extracting the next hop of the meta-paths obtained in the previous iteration, which are denoted as secondedge_set={AP, VP, KP, PA, PV, PK, PP}. Then the firstedge_set is updated by merging firstedge_set and secondedge_set. That is, the two-hops meta-paths are obtained, which are denoted as firstedge_set={PAP, PVP, PKP, PPA, PPV, PPK, PPP}. Obviously, the meta-paths with head node type P and tail node type V are now updated to IniPathSet= {PV, PPV}.

In the third iteration, the operations of the second iteration are repeated to obtain the three-hops meta-paths, so the firstedge_set

**Algorithm 1** AutoMP, a method for automatically searching initial meta-paths.

Input:
$T_G = (A, R)$; network schema of heterogeneous information network in adjacency matrix, where $A_i \in A$, $R_j \in R$
*Maxhop*; maximum length for meta-paths to be extracted
Output: $IniPathSet = \{P_1, P_2, ...\}$; set of meta-paths searched from $T_G$ in form of $A_m$ -*-$A_n$
/* auto-search initial meta-paths */
Begin
1: $IniPathSet \leftarrow Null$
2: for all $R_i \in R$, $R_i = (A_{im}, A_{in})$ do
3: $firstedge\_set \leftarrow list(A_{im})$ //create a list of edges starting from $A_{im}$
4: for $i \leftarrow 1$ to *Maxhop* do
5: $secondedge\_set \leftarrow next\_list(firstedge\_set)$ //Function 1
6: $merge\_set \leftarrow merge\_list(firstedge\_set, secondedge\_set)$ //Function 2
7: $firstedge\_set \leftarrow merge\_set$
8: for $j \leftarrow 1$ to $length(firstedge\_set)$ do
9: if last element of $firstedge\_set[j] == A_j$ then
10: $IniPathSet \leftarrow IniPathSet \cup firstedge\_set[j]$
11: end if
12: end for
13: end for
14: end for
End

is updated to firstedge_set={PAPA, PAPV, PAPK, PAPP, PVPA, PVPV, PVPK, PVPP, PKPA, PKPV, PKPK, PKPP, PPAP, PPVP, PPKP, PPPA, PPPV, PPPK, PPPP}. Then, IniPathSet is updated to IniPathSet={PV, PPV, PAPV, PVPV, PKPV, PPPV}.

Finally, in the fourth iteration, we can get IniPathSet={PV, PPV, PAPV, PVPV, PKPV, PPPV, PAPPV, PVPPV, PKPPV, PPPPC, PPAPV, PPVPV, PPKPV}.

The pseudo code of AutoMP is shown in Algorithm 1. It is worth noting that Function 1 and Function 2 are the same as those in the method proposed by Gupta, Kumar & Bhasker (2017). And the difference is that, AutoMP can search abundant meta-paths by utilizing multiple heterogeneous relation types, while the previous method only can generate meta-paths with the same type of head and tail nodes because it only utilizes one target relation type.

### 4.3. Generating node sequences

Most of the existing random walk methods for NRL are based on node-to-node transition probability, which is measured by only utilizing the direct topological relationship between the current node and the next hop node while ignoring the time information that may exist in networks, therefore the node sequences generated in this way cannot reflect the dynamics of networks. Meanwhile, those methods can only generate a varieties of node sequences with fixed length and fixed number, resulting in the generated node sequences cannot flexibly reflect the local and global characteristics of networks. However, the network structures of dynamic heterogeneous information networks are time-dependent and would change over time. In order to reflect the evolving characteristics of dynamic networks and construct optimized node sequences, we propose an Edge-based Dynamic Random Walk method (EDRW) to generate node sequences. EDRW calculates an edge-to-edge transition probability by utilizing the similarity between nodes within 2-hops and the timestamps of edges. In this way, not only the influence scope of each walk is expanded, but also the semantic information and dynamic structure information of the node sequences can be enhanced. In addition, in order to optimize node sequences and thus achieve better NRL performance, we propose two random walk control strategies based on the timestamp attributes of edges to dynamically adjust the length and number of node sequences.

Besides, although Metapath2vec (Dong, Chawla & Swami, 2017) can solve NRL problem on heterogeneous information networks, it learns the low-dimensional vectors of nodes only under the guidance of one or two specified symmetric meta-paths, resulting in the semantic information captured is simple. Meanwhile, Metapath2vec can only learn the representation of nodes in static networks and cannot be extended to dynamic networks. Therefore, in order to represent nodes more efficiently and accurately, we conduct EDRW on multiple initial symmetric and asymmetric meta-paths extracted by AutoMP method proposed in Section 4.2. In this way, different groups of node sequences are generated under different meta-paths, which ensures that more node types are sampled into node sequences and the node representation to be learned can reflect the heterogeneity of networks.

#### 4.3.1. Calculating edge-based dynamic transition probability

Fig. 3 shows the difference between a node-based random walk and the edge-based random walk proposed in this paper. The red dotted lines indicate the walking paths, and the black solid lines denote the existing edges between two nodes. In case of node-based random walk in Fig. 3(a), assuming that a walker has walked from node v1 to node v2, then the next hop of the walker is likely to be the node most relevant to v2, which is v3. However, in case of edge-based random walk in Fig. 3(b), the next hop is selected based on the reachability probability from both node v2 and node v1 which is the previous hop of v2. That is, the node most similar to both node v1 and node v2 will become the next hop, which is v4. Therefore, when the current node of the walker is v2, the next hop is different in node-based random walk and edge-based random walk.

Taking a meta-path (*APVPA*) extracted from the network schema in Fig. 2(a) as an example, we further analyze the difference between the above two random walk strategies. In case of edge-based random walk, when the node sequence under the guidance of sub-meta-path (*APV*) has been generated, the subsequent walk will be continued based on the transition probability to all *VP*-type edges from the given *PV*-type edge. That is, the most similar *VP*-type edge to the previous given *PV*-type edge is chosen under the guidance of sub-meta-path *APVP*, which is related to a 3-clique made up of these two edges. In comparison, in case of node-based random walk, when the node sequence under the guidance of sub-meta-path (*APV*) has been generated, the next *P*-type node may be any paper published on a previous given venue, without considering other papers related to the venue. It can be seen that edge-based random walk can construct node sequences with more similar nodes, which helps to reduce the generation of noisy node pairs.

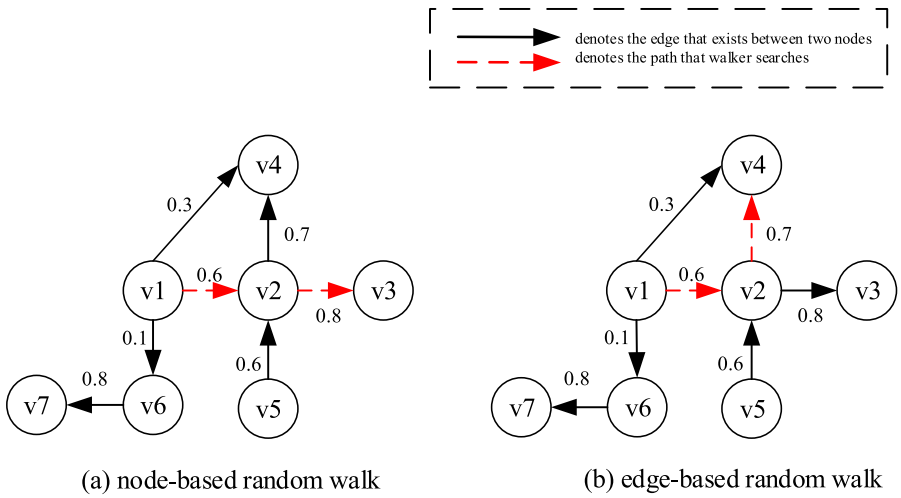



**Fig. 3.** Example of two different random walk strategies.

**Table 1**
Example of node-based transition probability matrix.

| | A | P | V | K |
|---|---|---|---|---|
| A | 0 | x | 0 | 0 |
| P | x | ✕ | x | ✕ |
| V | 0 | x | 0 | 0 |
| K | 0 | ✕ | 0 | 0 |

**Table 2**
Example of edge-based transition probability matrix.

| | AP | PA | PP | PV | VP | PK | KP |
|---|---|---|---|---|---|---|---|
| AP | 0 | ✕ | ✕ | x | 0 | ✕ | 0 |
| PA | ✕ | 0 | 0 | 0 | 0 | 0 | 0 |
| PP | 0 | ✕ | ✕ | ✕ | 0 | ✕ | 0 |
| PV | 0 | 0 | 0 | 0 | ✕ | 0 | 0 |
| VP | 0 | x | ✕ | ✕ | 0 | ✕ | 0 |
| PK | 0 | 0 | 0 | 0 | 0 | 0 | ✕ |
| KP | 0 | ✕ | ✕ | ✕ | 0 | ✕ | 0 |

In order to further explain the difference between node-based random walk and edge-based random walk, we also build a node-based transition probability matrix and an edge-based transition probability matrix under the meta-path *APVPA*, which are shown in Tables 1 and 2 respectively.

In Table 1, 0 means that there is no edge between two nodes, ✕ indicates two nodes are unreachable under the meta-path, and x indicates two nodes are reachable under the meta-path. In Table 2, 0 means that there is no hub node between two edges, ✕ indicates two edges are unreachable under the meta-path, and x indicates two edges are reachable under the meta-path. It can be seen that the edge-based random walk method considers more and higher order relationships than the node-based method, which helps to construct more accurate node sequences.

Following we discuss how to measure the dynamic transition probability of EDRW method.

On the one hand, we introduce the concepts of Center Type Node and Edge Type Node defined in Section 3 to measure the similarity between nodes.

For a center type node $v_i$, topic model is applied to extract its topic distributions, then its topic vector can be represented as:

$$Topic(v_i) = (w_{i1}, w_{i2}, ..., w_{iv}, ..., w_{ik}), \qquad \sum_{n=1}^{k} w_{in} = 1 \tag{1}$$

Where $k$ denotes the number of topics.

For an edge type node $v_j$ which is associated with one or more center type nodes, its latent topic information can be represented through the topic vectors of its associated center type nodes.

Therefore, we can measure the similarity between any types of nodes using their topic information.

On the other hand, considering a newly established relationship has more influence on the similarity between nodes while an old relationship has less influence on the similarity, we introduce a time decay function to reflect the difference.

Based on the above analysis, we distinguish the following three cases to calculate the similarity between two nodes.

(1) For two edge type nodes $N_a$ and $N_b$, the similarity is measured by utilizing the topic vectors of their associated center type nodes:

$$\begin{aligned} &s(N_a, N_b) \\ &= \cos\Big(\sum_{type(N_c)=Cnode,(N_c,N_a)\in E} \exp(-\eta(t_c - t_{N_c}))Topic(N_c), \\ &\quad \sum_{type(N_{c'})=Cnode,(N_{c'},N_b)\in E} \exp(-\eta(t_c - t_{N_{c'}}))Topic(N_{c'})\Big) \end{aligned} \tag{2}$$

Where $N_c$ and $N_{c'}$ denotes one of the center type nodes connected to nodes $N_a$ and $N_b$ respectively, $t_c$ denotes the current timestamp, $t_{N_c}$ denotes the timestamp of center type node $N_c$, $t_{N_{c'}}$ denotes the timestamp of center type node $N_{c'}$, $Topic(N_c)$ and $Topic(N_{c'})$ denotes the topic vectors of nodes $N_c$ and $N_{c'}$ respectively, and $\eta$ is a time decay factor which is set as an empirical value 0.62.

(1) For an edge type node $N_a$ and a center type node $N_c$, the similarity is calculated by utilizing the topic vector of the center type nodes associated with $N_a$ and the topic vector of $N_c$:

$$\begin{aligned} &s(N_a, N_c) = \cos\Big(\sum_{type(N_{c'})=Cnode,(N_{c'},N_a)\in E} \exp(-\eta(t_c - t_{N_{c'}})) \\ &Topic(N_{c'}), \exp(-\eta(t_c - t_{N_c}))Topic(N_c)\Big) \end{aligned} \tag{3}$$

(2) For two center type nodes $N_c$ and $N_{c'}$, the similarity is calculated by utilizing their respective topic vectors:

$$\begin{aligned} s(N_{c'}, N_c) = \cos(&\exp(-\eta(t_c - t_{N_{c'}}))Topic(N_{c'}), \\ &\exp(-\eta(t_c - t_{N_c}))Topic(N_c)) \end{aligned} \tag{4}$$

Then, the similarity between two edges is determined by the similarity between each two of the three nodes within two hops under a given meta-path:

$$s(r_i, r_{i-1}) = \begin{cases} s(A_{i-1}, A_{i+1}) + s(A_i, A_{i+1}) \\ \quad +s(A_{i-1}, A_i), & r_i \text{ is connected with } r_{i-1} \\ 0, & otherwise \end{cases} \tag{5}$$

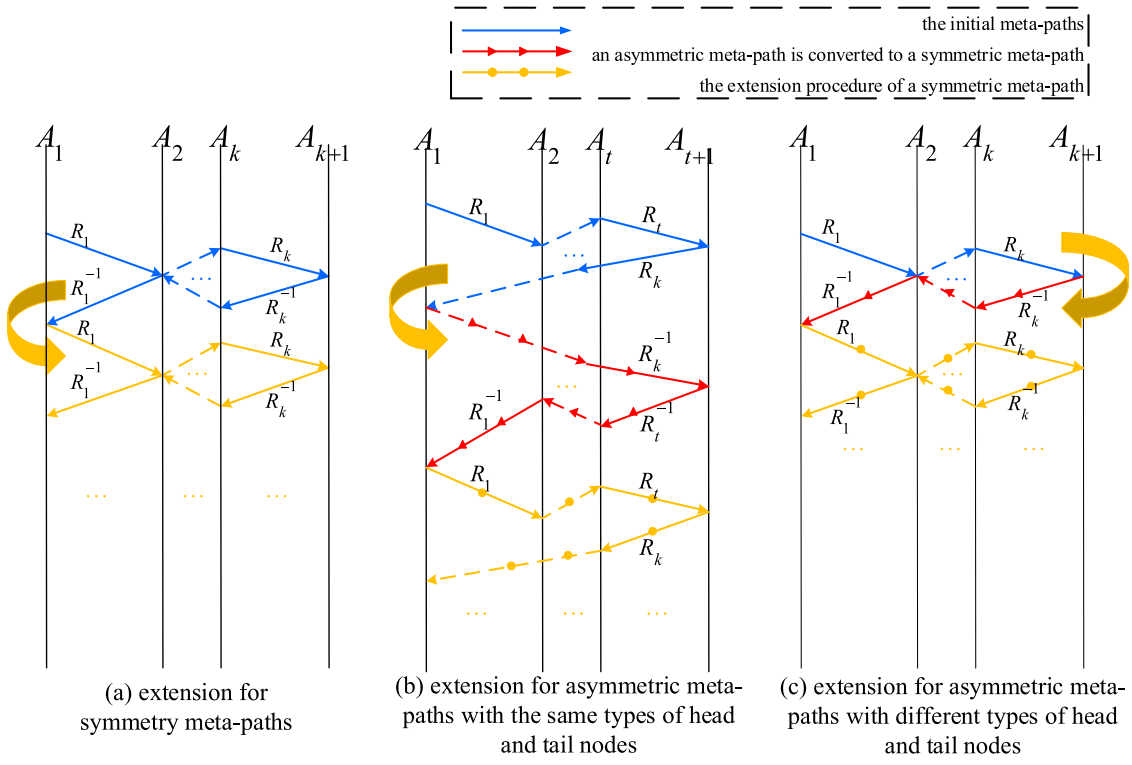



**Fig. 4.** Extension procedure of different types of initial meta-paths.

Where $s(A_{i-1}, A_{i+1})$ denotes the similarity between nodes $A_{i-1}$ and $A_{i+1}$, $A_i$ denotes the hub node of nodes $A_{i-1}$ and $A_{i+1}$, the type of $A_i$ is the $i^{th}$ node type under meta-path, and the type of $r_i$ is the $i^{th}$ relation type under meta-path.

Therefore, we define the transition probability in the $i^{th}$ step of edge-based random walk process as following, which incorporates the information of network structure, semantics, and timestamp.

$$p(r_i|r_{i-1}) = \begin{cases} \frac{\alpha*(s(r_i,r_{i-1}))+\beta}{\sum_{e\in NE(r_{i-1})}(\alpha*(s(r_{i-1},e))+\beta)}, & \\ \alpha+\beta=1, & r_i \text{ is connected with } r_{i-1} \\ 0, & otherwise \end{cases} \tag{6}$$

Where $\alpha$ denotes the weight of semantics, which indicates the similarity between edges measured using formula (5), $\beta$ denotes the weight of network structure, i.e., it indicates that the two edges are connected, $r_i$ denotes the edge appearing at the $i^{th}$ step of current random walk, whose type is the same as the $i^{th}$ relation type under the meta-path, $s(r_i,r_{i-1})$ denotes the similarity between edges $r_i$ and $r_{i-1}$, $NE(r_{i-1})$ denotes the neighbor edges that are connected with edge $r_{i-1}$ and the type of each edge in $NE(r_{i-1})$ is the same as $r_i$.

#### 4.3.2. Extending initial meta-paths

Considering that the initial meta-paths obtained by utilizing AutoMP described in Section 4.2 are limited in length, which are not sufficient for random walk to extract deeper relationships between nodes, while increasing their length can solve the problem (Fu, Lee and Lei, 2017), we suggest to extend the initial meta-paths.

Specifically, the extracted initial meta-paths are firstly converted to symmetric meta-paths, and then the symmetric meta-paths are repeatedly extended to support deeper guidance for random walk. For different types of meta-paths, including symmetric meta-paths, asymmetric meta-paths with the same types of head and tail nodes, and asymmetric meta-paths with different types of head and tail nodes, specific extension strategies are adopted in this paper.

As shown in Fig. 4, the blue arrow lines denote the initial meta-paths, the red arrow lines marked with triangles denote that asymmetric meta-paths are converted to symmetric meta-paths, and the yellow arrow lines marked with circles denote the extension procedure of symmetric meta-paths.

In Fig. 4(a), a symmetric meta-path $P = A_1 \xrightarrow{R_1} \cdots \xrightarrow{R_{k-1}} A_k \xrightarrow{R_k} A_{k+1} \xrightarrow{R_k^{-1}} A_k \xrightarrow{R_{k-1}^{-1}} \cdots \xrightarrow{R_1^{-1}} A_1$ is extended by a forward walk along the initial meta-path repeatedly until the walker stops. In this way, after the first extension operation, the initial meta-path is converted to $P = A_1 \xrightarrow{R_1} \cdots \xrightarrow{R_{k-1}} A_k \xrightarrow{R_k} A_{k+1} \xrightarrow{R_k^{-1}} A_k \xrightarrow{R_{k-1}^{-1}} \cdots \xrightarrow{R_1^{-1}} A_1 \xrightarrow{R_1} \cdots \xrightarrow{R_{k-1}} A_k \xrightarrow{R_k} A_{k+1} \xrightarrow{R_k^{-1}} A_k \xrightarrow{R_{k-1}^{-1}} \cdots \xrightarrow{R_1^{-1}} A_1$.

In Fig. 4(b), an asymmetry meta-path with the same types of head and tail nodes $P = A_1 \xrightarrow{R_1} \cdots \xrightarrow{R_{(t-1)}} A_t \xrightarrow{R_t} A_{t+1} \xrightarrow{R_k} A_k \xrightarrow{R_{(k-1)}} \cdots \xrightarrow{R_s} A_1$ is first converted to a symmetric meta-path $P = A_1 \xrightarrow{R_1} \cdots \xrightarrow{R_{(t-1)}} A_t \xrightarrow{R_t} A_{t+1} \xrightarrow{R_k} A_k \xrightarrow{R_{(k-1)}} \cdots \xrightarrow{R_s} A_1 \xrightarrow{R_s^{-1}} \cdots \xrightarrow{R_{(k-1)}^{-1}} A_k \xrightarrow{R_k^{-1}} A_{t+1} \xrightarrow{R_t^{-1}} A_t \xrightarrow{R_{(t-1)}^{-1}} \cdots \xrightarrow{R_1^{-1}} A_1$ by once reverse walk along the initial meta-path. Then, the strategy for symmetric initial meta-path in Fig. 4(a) is used for further extension.

In Fig. 4(c), an asymmetry meta-path with different types of head and tail nodes $P = A_1 \xrightarrow{R_1} \cdots \xrightarrow{R_{(k-1)}} A_k \xrightarrow{R_k} A_{k+1}$ is first converted to a symmetric meta-path $P = A_1 \xrightarrow{R_1} \cdots \xrightarrow{R_{(k-1)}} A_k \xrightarrow{R_k} A_{k+1} \xrightarrow{R_k^{-1}} A_k \xrightarrow{R_{(k-1)}^{-1}} \cdots \xrightarrow{R_1^{-1}} A_1$ by once reverse walk along the initial meta-path. Then, the strategy for symmetric initial meta-path in Fig. 4(a) is used for further extension.

Obviously, these extension operations do not add or reduce any semantic information of the initial meta-paths, and compared with the method of searching longer meta-paths first and then extracting the important ones, our method is more simple and effective.

In addition, as the extension operation happens along with the random walk process, the length of meta-paths used for semantic guidance is changeable in real time, so the random walk process can be flexibly guided to generate node sequences with variable length.

*4.3.3. Node sequences optimization strategies combining time information*

The existing node sequences generation methods based on random walk usually specify the length and number of node sequences, which makes the node sequences not flexibly reflect the hidden characteristics of networks (Nguyen et al., 2018). Nguyen et al. (2018) considered that random walk is related to the timestamps of edges, and each walker should walk in ascending order of timestamps of edges. However, their sampling proportion of the recently generated edges would be greatly reduced, leading to insufficient learning of the vertices of these edges. We believe that the timestamps of edges can be used to optimize the node sequences by both controlling the direction of random walk and dynamically adjusting the length and number of walk sequences. In addition, we think that the idea of Simulated Annealing algorithm can be used to control the stopping probability of each walker, so as to adjust the length of node sequences.

Based on the above considerations, we define several timestamp-related concepts and propose two timestamp-based random walk control strategies.

(1) **Several definitions of time-dependent terms**

**Definition 4.1 Long-term edge relationship and short-term edge relationship.** These two concepts are used to guide each step of walkers during random walk process. Specifically, given the timestamps $t_{e_i}$ and $t_{e_j}$ of edges $e_i$ and $e_j$, the timestamp distance $|t_{e_i} - t_{e_j}|$ is used to decide whether a relationship between two edges is a long-term edge relationship or a short-term edge relationship. That is, if $|t_{e_i} - t_{e_j}| < \varepsilon$, then $e_i$ has short-term edge relationship with $e_j$, otherwise they are long-term edge relationship with each other.

**Definition 4.2 Early edge and recent edge.** In order to reflect the evolving feature of dynamic networks, we distinguish the different effects of edges on network representation by their existing time and define two kinds of edges, called early edge and recent edge respectively. Specifically, given the occurrence timestamp $t_e$ of edge $e$ and the current timestamp of the network $t_c$, $|t_c - t_e|$ is used to decide whether edge $e$ is an early edge or a recent edge. That is, if $|t_c - t_e| < \varepsilon$, then edge $e$ is a recent edge, otherwise it is an early edge.

**Definition 4.3 Long-term event and short-term event.** A sequence of relationships only consisting of multiple short-term edge relationships is defined as a short-term event. If a relationship sequence contains one or multiple long-term edge relationships, it is defined as a long-term event.

Fig. 5 shows an example of the above concepts. Given a dynamic academic network in Fig. 5(a), we can obtain two different network schemas, respectively in Fig. 5(b) and Fig. 5(c). According to the network schema in Fig. 5(b), assuming that the threshold to distinguish long-term edge relationship and short-term edge relationship is 3, then "John-Paper1->Paper2-Mary" is a short-term event between John and Mary, because edges "John-Paper1" and "Paper1->Paper2" and edges "Paper1->Paper2" and "Paper2-Mary" are short-term edge relationships with each other. According to the network schema in Fig. 5(c), assuming that the current timestamp is 2019, and the threshold to distinguish recent edge and early edge is 3, then "John-Paper1" is a recent edge while "Mary-Paper2" is an early edge.

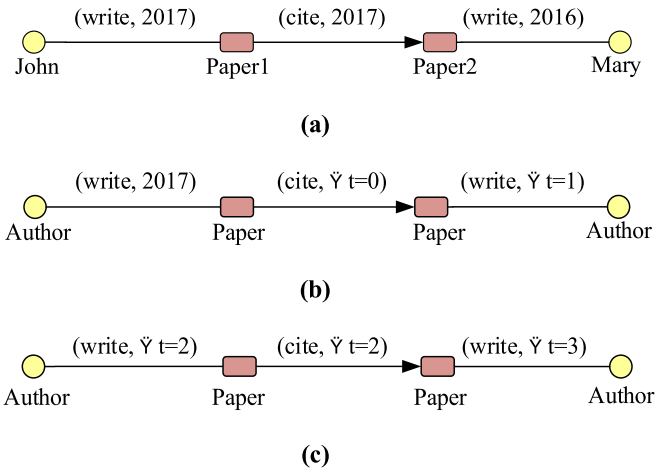



**Fig. 5.** An example of time-dependent terms. (a) A simple example of dynamic network structure. (b) Network schema based on source relationship in 2017. (c) Network schema based on current timestamp 2019.

Since only the timestamp information of center type nodes is provided in the source dataset, we have to obtain the timestamps of edges from the associated center type nodes for the following strategies. For example, the timestamp of edge AP is the same as the publishing time of its associated center type node P.

(2) **Length optimization strategy for node sequences (LOS)**

We optimize the length of node sequences by distinguishing long-term edge relationship and short-term edge relationship during each random walk process. In our opinion, short-term edge relationship has larger impact on a target edge while long-term edge relationship has less impact. Therefore, we propose that short-term edge relationship should be sampled more than long-term edge relationship when generating node sequences. When a long-term edge relationship appears in a node sequence, the stopping probability of current walker should be appropriately increased to avoid walk bias. However, long-term edge relationship can enrich the long-term characteristics of node sequences. Taking the above factors into account, we adopt the idea of Simulated Annealing to determine a stopping probability for each walker, so as to dynamically adjust the length of the node sequences generated by random walk.

Specifically, we set a stopping probability individually for each walker. If the distance between the timestamps of the next edge $e$ and the starting edge is smaller than a given threshold $\varepsilon$ which is set to 3 in this paper, these two edges are in a short-term edge relationship, then the walker moves to edge $e$ at a probability of 1. Otherwise, they are in a long-term edge relationship, and the walker should stop with a probability of $\gamma$, whose value increases as the number of long-term edge relationships during each walk increases and it is defined as:

$$\gamma = e^{-a/(x+b)}, \qquad a > 0, b > 0, x > 0 \tag{7}$$

Where $x$ denotes the number of long-term edge relationships that have appeared during the random walk so far, $a$ is a time decay factor which is used to reduce the occurrence probability of long-term edge relationships, and it is set to an empirical value of 0.62, $b$ is used to smooth the value of $x$, and its value is set to 1.

As a result of the above LOS strategy, the sampling rate of more similar nodes will be increased while that of the less similar ones will be decreased during the generation of each node sequence, which ensuring that the similar node pairs are closer while the non-similar ones are farther in a low-dimensional space.

(3) **Quantity optimization strategy for node sequences (QOS)**

We optimize the number of node sequences by distinguishing early edge and recent edge at the beginning of each random walk process. Generally, information networks contain fewer recent edges and more early edges. Meanwhile, more short-term edge relationships happen between early edges. Therefore, if uniform sampling strategy is adopted during a random walk process, a large number of early edges would be sampled, resulting that they occupy the majority of sampling space after a random walk process, and the node representation would be more influenced by early edges rather than recent edges.

Based on the above considerations, we propose that the number of walkers should be determined by the existing time of starting edges so as to optimize the number of node sequences generated through random walk process. Specifically, we increase the sample number of recent edges and reduce the sample number of early edges to enhance the effect of recent edges and weaken the effect of early ones. Here, $\lambda$ and $\varsigma$ denote the sample number of early edges and recent edges respectively, which satisfy $\varsigma = 2\lambda$.

#### 4.3.4. EDRW algorithm

EDRW consists of two steps: (1) constructing an edge-based transition probability matrix according to the measure method proposed in Section 4.3.1; (2) performing meta-path guided random walk by combining the two optimization strategies proposed in Section 4.3.3.

The pseudocode of EDRW is shown in Algorithm 2.

### 4.4. Learning the low-dimensional node representation

The experimental results in Metapath2vec (Dong, Chawla & Swami, 2017) show that the heterogeneous Skip-Gram model can effectively learn the features of different nodes since it distinguishes the heterogeneous context of different nodes and utilizes a heterogeneous negative sampling method to learn node representation. The objective of heterogeneous Skip-Gram is to maximize the heterogeneous context information of given nodes, that is:

$$\arg\max_{\theta} \sum_{v\in V} \sum_{t\in T_V} \sum_{c_t\in N_t(v)} \log p(c_t|v;\theta) \tag{8}$$

Where $N_t(v)$ denotes the $t^{th}$ neighbor node of node $v$, and $p(c_t|v;\theta)$ denotes a softmax function as,

$$p(c_t|v;\theta) = e^{X_{ct}\cdot X_v} / \sum_{u_t\in V_t} e^{X_{ut}\cdot X_v} \tag{9}$$

Where $X_v$ is the row of $X$, which denotes the low dimensional vector of node $v$, and $V_t$ denotes the $t^{th}$ node type in the network.

Finally, in order to achieve efficient optimization, negative sampling technique introduced by Mikolov et al. (Mikolov et al., 2013) is leveraged and modified to be heterogeneous by Dong et al., so the objective function of heterogeneous Skip-Gram becomes as:

$$\begin{aligned} O(X) &= \log p(c_t|v;\theta) = \log \sigma(X_{c_t}\cdot X_v) \\ &\quad + \sum_{m=1}^{M} E_{u_t^m\sim P_t(u_t)}\left[\log \sigma(-X_{u_t^m}\cdot X_v)\right] \end{aligned} \tag{10}$$

Where $\sigma(x) = 1/(1+e^{-x})$, and $M$ is the number of negative samples whose distribution is $P(u)\propto d_{out}(u)^{3/4}$ (Mikolov et al., 2013).

### 4.5. Time complexity analysis

The time complexity of HIN_DRL mainly comes from generating node sequences and training Skip-Gram model. Since the time complexity of heterogeneous Skip-Gram model is analyzed (Dong, Chawla & Swami, 2017), here we only analyze the part of generating node sequences as following.

Firstly, we obtain the topic distribution of each center type node by utilizing the LDA model, and the corresponding time complexity is $O(knV_C\bar{l})$, where $k$ is the number of topics, $n$ is the number of iterations, $V_C$ is the number of center type nodes, and $\bar{l}$ is the average text length contained in $V_C$. The time complexity of this part is equivalent to $O(v)$, where $v$ is the number of nodes in network.

Then, we calculate the similarity between nodes by utilizing topic vectors, the corresponding time complexity is $O(e)$, where $e$ is the number of edges in network. Next, the similarity between edges is calculated, whose time complexity is $O(V_{H_i}\cdot d_{H_i\cdot H_{i-1}}\cdot d_{H_i\cdot H_{i+1}})$, where $V_{H_i}$ denotes the number of hub nodes, $d_{H_i\cdot H_{i-1}}$ and $d_{H_i\cdot H_{i+1}}$ denotes the number of the previous and subsequent neighbor nodes of hub nodes respectively. The time complexity of this part is $O(V_{H_i}) \approx O(e^2)$.

Finally, we get the normalized transition probability, whose time complexity is $O(e^2)$. Given the transition probability, we can perform edge-based random walk process, whose time complexity is $O(\bar{n}e^2)$, where $\bar{n}$ is the average number of walkers leading by starting edges.

Therefore, the total time complexity for generating node sequences based on our proposed method is $O(knV_C\bar{l} + V_{H_i}\cdot d_{H_i\cdot H_{i-1}}\cdot d_{H_i\cdot H_{i+1}} + \bar{n}e^2 + e^2 + e)$, which is equivalent to $O(e^2)$. It is worth noting that parallel processing can be applied to speed up the calculation, for example, random walkers guided by different meta-paths can be executed simultaneously.

### 4.6. Space complexity analysis

Similarly, here we only discuss the space complexity of HIN_DRL before training Skip-Gram model, which mainly comes from constructing dynamic heterogeneous information network and generating node sequences.

Firstly, we design the network schema and build the target network using the entities, relations of the being learnt network. The corresponding space complexity is $O(C + v + kV_c + e)$, where $C$ is the number of entity types and relation types in the network schema which is a constant, $V_C$ is the number of center type nodes, $k$ is the size of topic vector for $V_C$, $v$ and $e$ are the number of nodes and edges in network separately. Thus, the space complexity of this stage is $O(C + v + kV_c + e) \approx O(v + e) \approx O(n)$, where $n = max(v, e)$.

Then, we generate node sequences by utilizing transition probability matrix, the space complexity of the corresponding matrix is $O(e + e^2)$, where only the similarity between the directly connected nodes and that between the directly connected edges are calculated.

Finally, node sequences are collected, whose space complexity is $O(\bar{l}\bar{n}e)$, where $\bar{l}$ denotes the average length of all node sequences starting by edges, and $\bar{n}$ denotes the average number of walkers guiding by the starting edges. So the space complexity of this part is $O(e + e^2 + \bar{l}\bar{n}e) \approx O(e^2)$.

Therefore, the total space complexity is $O(C + v + kV_c + e + e + e^2 + \bar{l}\bar{n}e)$, which is equivalent to $O(e^2)$.

**Algorithm 2** EDRW (Edge-based Dynamic Random walk).

Input:
$G_T = (V, E, A, R, T)$, dynamic heterogeneous information network created under current timestamp $t_c$
$IniPathSet = \{P_1^{ini}, P_2^{ini}, \ldots\}$, set of initial meta-paths extracted from $T_G$ in form of $A_i$ -*-$A_j$
*MaxLen*, the longest length of the node lists to be generated
$\lambda$, $\varsigma$, the number of walkers starting from early edges and recent edges respectively
Output:
*Nodelist*= $\{NL_1, NL_2, \ldots\}$, set of node lists in form of $a_i$-$a_{i+1}$*-$a_k$
/* generate node lists */
Begin
/* calculate the edge-based transform probability */
1: for all $v_i$, $v_j \in V$ do //calculate the similarity between nodes
2: if there is an edge between $v_i$ and $v_j$ *then*
3: calculate $sim(v_i, v_j)$ according to formula (2)-(4)
4: end if
5: end for
6: for *all* $e_i$, $e_j \in E$ do //calculate the similarity between edges
7: if these is a hub node between edges $e_i$ and $e_j$ *then*
8: calculate $sim(e_i, e_j)$ according to formula (5)
9: end if
10: end for
11: calculate the edge-based transform probability according to formula (6)
/* generate node lists through edge-based random walk */
12: *Nodelist*←*Null*
13: for all $P_k^{ini} \in IniPathSet$ do
14: $j \leftarrow 1$ // $j$ is used to indicate which relation type should be used for next walk step along a meta-path
15: convert $P_k^{ini}$ to $P_k^{sym}$ according to 4.3.2
16: *tmpSequences*←select $e_t$ where $\psi(e_t) == R_j$ // find edges with the 1st relation type of $P_k^{sym}$
17: $j \leftarrow j+1$
18: *firstSequences*←*RW_first_list(tmpSequences, λ, ς) //function call*
19: for $i \leftarrow 1$ to *MaxLen* do
20: *secondSequences*←RW_*next_list*(*firstSequences*,$P_k^{sym}$, *i*, *j*) //function call
21: *mergeSequences*←RW_*merge_list*(*firstSequences, secondSequences, i*) //function call
22: *firstSequences*←*mergeSequences*
23: if *i* is integer multiples of the length of $P_k^{sym}$ then
24: $j \leftarrow 1$ // back to the first relation in $P_k^{sym}$
25: else
26: $j \leftarrow j+1$
27: end if
28: end for
29: *Nodelist*←*Nodelist*∪*firstSequences*
30: end for
End

Function3: RW_first_list, determine the first edge of the walker

*RW_first_list (tmpSequences, λ, ς)*
*{*
*1: firstSequences*←*Null*
*2:* for all $e_t \in tmpSequences$ do
*3:* if $e_t$ is an early edge then // according to QOS in Section 4.3.3
*4:* for $i \leftarrow 1$ to λ do
*5:* *firstSequences*←*firstSequences*∪$e_t$
*6:* end for
*7:* else if $e_t$ is a recent edge then // according to QOS in 4.3.3
*8:* for $i \leftarrow 1$ to ς do
*9: firstSequences*←*firstSequences*∪$e_t$
*10:* end for
*11:* end if
*12:* end for
*13:* return*(firstSequences)*
*}*

Function4: RW_next_list, determine the next edge of the walker

*RW_next_list(firstSequences,* $P_k^{sym}$, *n, m)*
*{*
1: *secondSequences*←*Null, nodes*←*Null,*
2: *firstSequences*←select $f_t$ where *length*($f_t$)== *n* and $f_t \in firstSequences$
3: for $i \leftarrow 1$ to *length(firstSequences)* do
4: *nodes*←*nodes*∪last element of *firstSequences[i]*
5: end for
6: for $i \leftarrow 1$ to *length(nodes)* do
7: *neb*←find neighbors with the $(m+1)^{th}$ node type under $P_k^{sym}$ of *nodes[i]* and choose one neighbor according to formula (6)
9: *secondSequences*←*secondSequences*∪*concatenate(nodes[i],neb)*
10: end for

(*continued on next page*)

**Algorithm 2** (*continued*)

11: return(*secondSequences*)
}
Function5: RW_ merge_list, merge firstSequences and secondSequences

*RW_merge_list(firstSequences, secondSequences, n)*
{
1: *mergeSequences*←*Null*
2: for *i*←*1* to *length(firstSequences)*, *j*←*1* to *length (secondSequences)* do
3: if *length (firstSequences[i])* is *n* and the last element of *firstSequences[i]* is the same as the first element of *secondSequences[j]* then
4: if the first edge of *firstSequences[i]* and *secondSequences[j]* is short-term-edge relationship then
5: *tmp*←*concatenate (firstSequences[i], secondSequences[j])*
6: else if the first edge of *firstSequences[i]* and *secondSequences[j]* is long-term-edge relationship then
7: calculate the stop probability $\gamma$ according to formula (7) in 4.3.3.1(LOS)
8: if$\gamma$==1 then
9: *tmp*←*firstSequence[j]*
10: else
11: do *tmp*←*concatenate(firstSequences[i], secondSequences[j])* with probability=1-$\gamma$
12: end if
13: end if
14: else
15: *tmp*←*firstSequences[j]*
16: end if
17: *merge_set*←*merge_set*∪*tmp*
18: end for
19: return(*mergeSequences*)
}

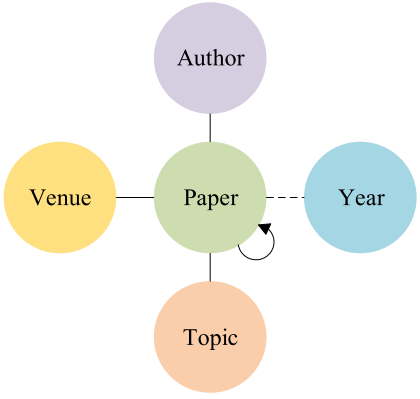



**Fig. 6.** Network scheme for Microsoft Academic.

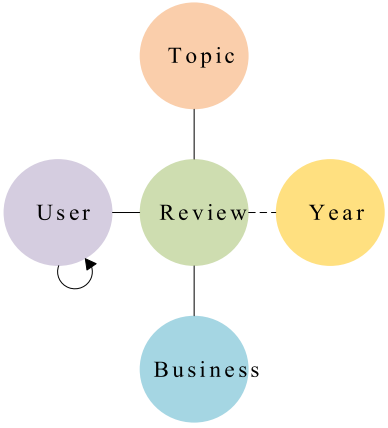



**Fig. 7.** Network scheme for YELP.

## 5. Experiments and results analysis

### 5.1. Experimental setup

#### 5.1.1. Datasets

In this paper, Microsoft Academic dataset, YELP dataset and IMDB dataset are used to verify the representational effects of HIN_DRL on different types of heterogeneous information networks.

Microsoft Academic dataset is obtained from Microsoft Academic[1], which covers the publication information about the field of Computer Science from years 2001 to 2010, including 112,358 authors, 57,446 papers and 6,319 venues. Meanwhile, in order to make use of the text information, such as title, abstract and keywords of papers, we construct a 100-dimensional topic vector for each paper node by utilizing LDA model. The network schema for Microsoft Academic dataset is specified as Fig. 6, which includes four types of entities, namely author, paper, venue (journal/conference) and topic, and four types of relationships, namely paper-author, paper-paper, paper-venue and paper-topic. The dotted line in Fig. 6 denotes the timestamp attribute of paper entity.

YELP dataset is officially provided by YELP website[2], which covers the review data from years 2004 to 2012, including 121,670 reviews, 6,908 businesses and 32,382 users. Same as Microsoft Academic Dataset, we obtain 100 topics from the short-text of the reviews. The network schema for YELP dataset is shown in Fig. 7, which includes four types of entities, namely review, business, user and topic, and four types of relationships, namely review-business, review-user, user-user and review-topic. The dotted line in Fig. 7 denotes the timestamp attribute of review entity.

IMDB dataset is crawled from IMDB's official website[3], which covers the movie data released from years 1916 to 2014, including 4,500 movies, 34,570 actors/actress, and 2,245 directors. We also extract 100 topics from the storyline of the movies. The network schema of IMDB dataset is shown in Fig. 8, which includes four types of entities, namely movie, director, actor/actress and topic, and three types of relationships, namely movie-director, movie-

[1] http://academic.research.microsoft.com. Accessed on July, 2015.

[2] https://www.yelp.com/dataset. Accessed on March, 2018.

[3] https://www.imdb.com. Accessed on January, 2015.

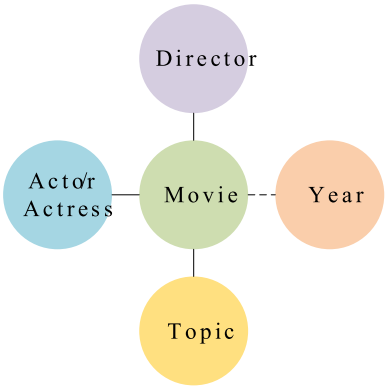



**Fig. 8.** Network scheme for IMDB.

actor/actress and movie-topic. The dotted line in Fig. 8 denotes the timestamp attribute of movie entity.

*5.1.2. Baselines*

Firstly, among those NRL algorithms designed for homogeneous information networks, two random walk based methods including DeepWalk (Perozzi, Al-Rfou & Skiena, 2014) and Node2vec (Grover & Leskovec, 2016) and the edge reconstruction based method LINE (Tang et al., 2015a) are selected as part of the baselines. Then, among the NRL algorithms for heterogeneous information networks, Metapath2vec and Metapath2vec++ (Dong, Chawla & Swami, 2017), which are closest to our algorithm, are selected as the other baseline methods. These baseline methods are described below:

(1) DeepWalk. It is the first proposed classical algorithm based on random walk. The algorithm can only be applied to homogeneous information networks, where node sequences are constructed by uniform random walk and node representations are learnt by Skip-Gram model.
(2) Node2vec. Unlike DeepWalk, Node2vec can flexibly mine network structure information by combining BFS and DFS strategies to conduct random walk process. The algorithm uses two parameters $p$ and $q$ to adjust the weights of BFS and DFS. When $p$=1 and $q$=1, it is generally considered to be equivalent to DeepWalk. In the following experiments, we adopt grid search method over $p, q \in \{0.25, 0.5, 1, 2, 4\}$ as mentioned in the previous work (Grover & Leskovec, 2016).
(3) LINE. This algorithm models the relationships between nodes by considering first-order proximity and second-order proximity, constructs a loss function by using the difference between the theoretical probability distribution and the empirical probability distribution of two nodes on one edge, and then obtains the low-dimensional vectors of nodes by minimizing the loss function.
(4) Metapath2vec and Metapath2vec++. In order to mine the heterogeneity of networks, these two methods generate node sequences by utilizing meta-path based random walk process, in which node types are no longer ignored, and network semantic information can be learnt. After generating node sequences, a new heterogeneous Skip-Gram model is used for both Metapath2vec and Metapath2vec++, and a new heterogeneous negative sampling method is further used for Metapath2vec++, which can distinguish different types of neighbor nodes for representation learning.

*5.1.3. Parameter settings*

Except LINE, the algorithms based on random walk need to set some general parameters as following:

(1) Number of walkers starting random walk from each node ($w$=1000)
(2) Maximum length of per walk ($l$=100)
(3) Dimensions of low-dimensional vector ($d$=128)
(4) Window size of the neighbor nodes that is delineated ($k$=7)
(5) The number of negative samples ($n$=5)

It is noted that parameter$w$is only used for the methods of node-based random walk, including DeepWalk, Node2vec, Metapath2vec and Metapath2vec++. In HIN_DRL, two other parameters are designed to provide similar function like$w$, namely$\lambda$and$\varsigma$in Section 4.3.3. In following experiments, we set $\lambda$=1000, $\varsigma$=2000.

Besides, we adopt additional five groups of parameters in HIN_DRL, described as following.

(1) $\eta$: This is a time decay factor used to calculate the similarity between nodes in (2)–(4) of Section 4.3.1, which makes early edges less important and recent edges more important and is set as an empirical value 0.62.
(2) $\alpha$ and $\beta$: These two parameters are used to trade-off the weight of semantic information and network structure information when calculating the transition probability between edges in (6) of Section 4.3.1. Here, it is assumed that semantic information is more important, and we set $\alpha$=0.9, $\beta$=0.1.
(3) $\varepsilon$: This parameter is used for distinguishing short-term/long-term edge relationship and early/recent edge in Section 4.3.3. In following experiments, we set $\varepsilon$=3.
(4) $a$ and $b$: These two parameters are used for calculating the stopping probability $\gamma$ of a walker in EDRW according to (8) in Section 4.3.3, whose value increases when a long-term edge relationship appears. In this paper, we set $a$=0.62, and $b$=1.
(5) $t_c$: This is used for determining the current network state. In this paper, the current timestamp is always set as the latest timestamp of a dataset. That is, the current timestamp is set as 2010 in Microsoft Academic dataset, 2012 in YELP dataset, and 2014 in IMDB dataset.

*5.1.4. Experimental schemes*

The following machine learning tasks on three datasets are used to verify the effects of our proposed HIN_DRL method and the baseline methods. Since the datasets we get are different, we perform different tasks on them. In which, each tagged node has just one label in Microsoft Academic dataset, and each tagged node has one or more labels in YELP dataset and IMDB dataset. The first three of the following tasks are implemented by utilizing the sklearn package in machine learning Library of Python.

(1) Multi-class node classification: This task is performed on Microsoft Academic dataset.
(2) Multi-label node classification: This task is performed on YELP dataset and IMDB dataset.
(3) Node clustering: This task is performed using FCM algorithm (Bezdek, 1981) for YELP dataset and IMDB dataset, and k-means algorithm for Microsoft Academic dataset.
(4) Visualization: We visualize the experimental results of HIN_DRL and the baseline methods on Microsoft Academic dataset via 2-D visualization graph.
(5) Dynamics of HIN_DRL: We first learn the different node representations for Microsoft Academic network on two different current timestamps of year 2004 and year 2010 by utilizing HIN_DRL, then analyze the evolution characteristics of the network via 2-D visualization graph.

*5.2. Experimental results and discussions*

*5.2.1. Node classification*

We first verify the effects of HIN_DRL method on multi-class node classification and multi-label node classification tasks. These

**Table 3**
Multi-class node classification of venue nodes in Microsoft Academic dataset (Macro-F1).

| Training set size | 5% | 10% | 20% | 30% | 40% | 50% | 60% | 70% | 80% | 90% | Mean Macro-F1 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| DeepWalk | 0.140 | 0.204 | 0.351 | 0.445 | 0.502 | 0.541 | 0.581 | 0.607 | 0.612 | 0.654 | 0.463 |
| LINE | 0.062 | 0.084 | 0.104 | 0.143 | 0.193 | 0.193 | 0.213 | 0.224 | 0.227 | 0.240 | 0.168 |
| Node2vec | 0.141 | 0.210 | 0.357 | 0.452 | 0.516 | 0.546 | 0.588 | 0.619 | 0.628 | 0.664 | 0.472 |
| Metapath2vec | 0.151 | 0.220 | 0.405 | 0.477 | 0.540 | 0.545 | 0.579 | 0.603 | 0.619 | 0.687 | 0.482 |
| Metapath2vec++ | **0.173** | **0.279** | 0.390 | 0.470 | 0.537 | 0.541 | 0.578 | 0.620 | 0.630 | 0.675 | 0.485 |
| HIN_DRL | 0.172 | 0.271 | **0.409** | **0.485** | **0.553** | **0.584** | **0.615** | **0.635** | **0.658** | **0.708** | **0.509** |

two node classification tasks need to use the label information of network nodes, which can be obtained from the source dataset for YELP and IMDB networks. However, considering that Microsoft Academic dataset does not contain label information of network nodes, we refer to the method used in Metapath2vec to determine the node label by utilizing third-party labels. Specifically, we select 10 categories[4] (top 12 venues are selected for each category) in the field of Computer Science from Google Scholar[5]. Among all the 120 venues, 103 venues of them are matched and labeled correspondingly.

(1) **Multi-class node classification**

Multi-class node classification task is used to evaluate the effects of different NRL methods for case that the whole dataset (i.e., Microsoft Academic dataset in our experiments) has multiple labels but each node in the dataset is assigned to only one label, and Macro-F1 score and Micro-F1 score (Manning, Raghavan & Schütze, 2008) are used for evaluation criteria.

In detail, Macro-F1 score is defined as,

$$Macro - F1 = \frac{\sum_{l \in L} F1(l)}{|L|} \tag{11}$$

Where $F1(l)$ denotes the F1-score of label $l$, $|L|$ denotes the size of the label set.

Micro-F1 score is defined as,

$$Micro - F1 = \frac{2 * P * R}{P + R},$$
$$P = \frac{\sum_{l \in L} TP(l)}{\sum_{l \in L} (TP(l) + FP(l))}, \; R = \frac{\sum_{l \in L} TP(l)}{\sum_{l \in L} (TP(l) + FN(l))} \tag{12}$$

Where $TP(l)$, $FP(l)$ and $FN(l)$ are the True Positive, False Positive and False Negative of label $l$ respectively.

In this paper, the logistic regression classifier in the multi-class classifier provided by the sk-learn package in machine learning library of Python is used for this classification task. We first learn the low-dimensional representation of network nodes using the entire dataset. Then, the learned low-dimensional representation is input into the classifier, where 5% to 90% of them is randomly selected as the training set for multiple experiments respectively, each experiment is repeated 10 times, and the average Macro-F1 and Micro-F1 for each experiment are reported.

Tables 3 and 4 and Figs. 9 and 10 respectively show the Macro-F1 scores and Micro-F1 scores of multi-class node classification for Microsoft Academic dataset.

As can be seen from Tables 3 and 4, among all algorithms, LINE performs the worst, whose average Macro-F1 score is less than 0.2 and the average Micro-f1 score is less than 0.3; while HIN_DRL performs the best. In term of average Macro-F1 scores, HIN_DRL achieves 2.44% (relatively 5.0%) improvement over Metapath2vec++, 2.65% (relatively 5.5%) over Metapath2vec, 3.70% (relatively7.8%) over Node2vec, 4.55% (relatively 9.8%) over DeepWalk and 34.10% (relatively 6.5%) over LINE. In term of average Micro-F1 scores, HIN_DRL achieves 2.08% (relatively 3.5%) gain over Metapath2vec++, 3.74% (relatively 6.5%) over Metapath2vec, 4.10% (relatively 7.2%) over Node2vec, 5.72% (relatively 10.4%) over DeepWalk, and 35.78% (relatively 142.2%) over LINE.

It can also be seen from Figs. 9 and 10 that, except for LINE, the Macro-F1 scores and Micro-F1 scores of all other methods increase faster with the increase of training set, and HIN_DRL achieves the

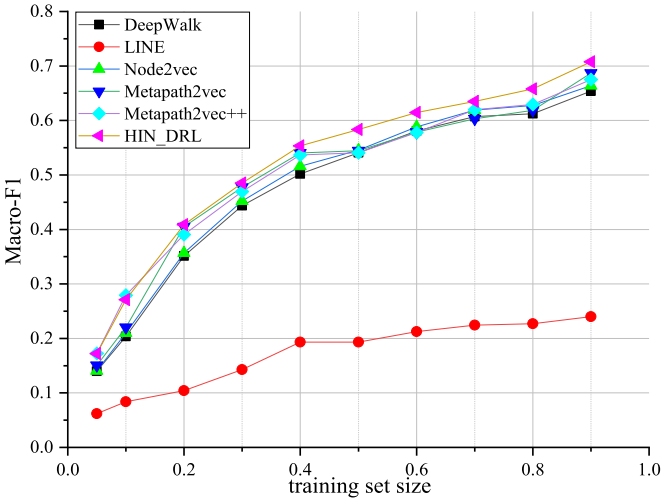



**Fig. 9.** Multi-class node classification of venue nodes in Microsoft Academic dataset (Macro-F1).

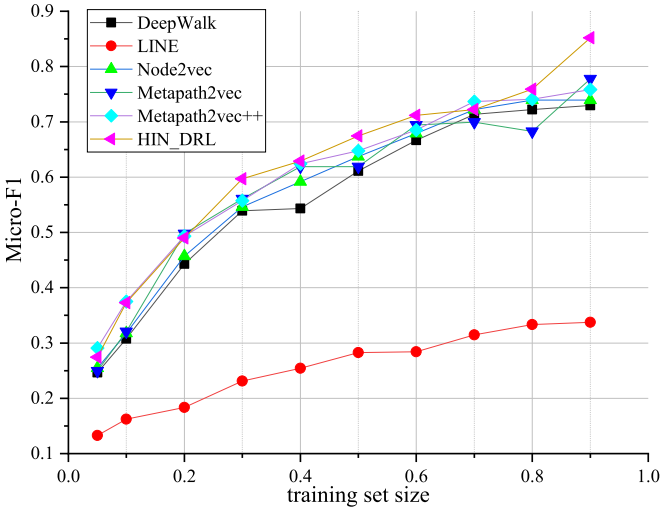



**Fig. 10.** Multi-class node classification of venue nodes in Microsoft Academic dataset (Micro-F1).

[4] Artificial Intelligence, Computational Linguistics, Computer Graphics, Computer Networks & Wireless Communication, Computer Vision & Pattern Recognition, Computing Systems, Data Mining & Analysis, Databases & Information Systems, Human Computer Interaction, and Theoretical Computer Science.

[5] https://scholar.google.com/citations?view op=top venues&hl=en&vq=eng. Accessed on June, 2018.

**Table 4**
Multi-class node classification of venues in Microsoft Academic dataset (Micro-F1).

| Training set size | 5% | 10% | 20% | 30% | 40% | 50% | 60% | 70% | 80% | 90% | Mean Micro-F1 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| DeepWalk | 0.246 | 0.308 | 0.443 | 0.539 | 0.543 | 0.611 | 0.667 | 0.714 | 0.722 | 0.730 | 0.552 |
| LINE | 0.133 | 0.162 | 0.184 | 0.231 | 0.254 | 0.283 | 0.284 | 0.315 | 0.333 | 0.338 | 0.252 |
| Node2vec | 0.255 | 0.318 | 0.457 | 0.546 | 0.592 | 0.637 | 0.679 | 0.722 | 0.739 | 0.739 | 0.568 |
| Metapath2vec | 0.250 | 0.321 | **0.498** | 0.561 | 0.619 | 0.619 | 0.694 | 0.700 | 0.683 | 0.778 | 0.572 |
| Metapath2vec++ | **0.291** | **0.375** | 0.493 | 0.557 | 0.624 | 0.648 | 0.685 | **0.737** | 0.741 | 0.758 | 0.589 |
| HIN_DRL | 0.274 | 0.373 | 0.490 | **0.597** | **0.629** | **0.675** | **0.712** | 0.722 | **0.759** | **0.852** | **0.609** |

**Table 5**
Multi-label node classification of movie nodes in IMDB dataset (Macro-F1).

| Training set size | 5% | 10% | 20% | 30% | 40% | 50% | 60% | 70% | 80% | 90% | Mean Macro-F1 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| DeepWalk | 0.159 | 0.197 | 0.208 | 0.216 | 0.222 | 0.232 | 0.234 | 0.241 | 0.245 | 0.246 | 0.220 |
| LINE | 0.119 | 0.125 | 0.133 | 0.133 | 0.141 | 0.142 | 0.143 | 0.144 | 0.145 | 0.149 | 0.137 |
| Node2vec | 0.167 | 0.192 | 0.209 | 0.215 | 0.222 | 0.231 | 0.241 | 0.244 | 0.255 | 0.255 | 0.223 |
| Metapath2vec | 0.185 | 0.199 | 0.217 | 0.224 | 0.231 | 0.236 | 0.238 | 0.244 | 0.253 | 0.258 | 0.228 |
| Metapath2vec++ | 0.184 | 0.200 | 0.216 | 0.229 | 0.233 | 0.239 | 0.241 | 0.249 | 0.260 | 0.262 | 0.231 |
| HIN_DRL | **0.191** | **0.216** | **0.223** | **0.235** | **0.249** | **0.251** | **0.259** | **0.261** | **0.269** | **0.272** | **0.243** |

**Table 6**
Multi-label node classification of movie nodes in IMDB dataset (Micro-F1).

| Training set size | 5% | 10% | 20% | 30% | 40% | 50% | 60% | 70% | 80% | 90% | Mean Micro-F1 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| DeepWalk | 0.364 | 0.368 | 0.386 | 0.418 | 0.439 | 0.450 | 0.462 | 0.475 | 0.475 | 0.481 | 0.432 |
| LINE | 0.362 | 0.367 | 0.373 | 0.376 | 0.380 | 0.381 | 0.386 | 0.386 | 0.386 | 0.389 | 0.378 |
| Node2vec | 0.363 | 0.369 | 0.383 | 0.412 | 0.430 | 0.457 | 0.467 | 0.476 | 0.476 | 0.481 | 0.431 |
| Metapath2vec | 0.375 | 0.386 | 0.414 | 0.435 | 0.450 | 0.457 | 0.465 | 0.476 | 0.478 | 0.485 | 0.442 |
| Metapath2vec++ | 0.369 | 0.388 | 0.413 | 0.439 | 0.452 | 0.464 | 0.469 | 0.476 | 0.482 | 0.487 | 0.444 |
| HIN_DRL | **0.381** | **0.402** | **0.436** | **0.458** | **0.471** | **0.481** | **0.490** | **0.496** | **0.498** | **0.506** | **0.462** |

best effects under most of the different training dataset sizes. The above results show that both the semantic information and time information in academic networks improve the effects of HIN_DRL on multi-class node classification task.

(2) **Multi-label node classification**

Multi-label node classification task is used to evaluate the effects of different NRL methods for the dataset where each node is assigned to one or more labels (i.e., YELP dataset and IMDB dataset in this paper), which is performed using the logistic regression classifier of the multi-label classifier provided by the sk-learn package of Python machine learning library. Similarly, we first learn the low-dimensional node representation by utilizing the entire dataset. Then, the learned low-dimensional representation is input into the classifier, in which 5% to 90% of them is randomly selected as the training set for multiple experiments respectively, each experiment is repeated 10 times and the average Macro-F1 scores and Micro-F1 scores are reported finally.

Tables 5 and 6 and Figs. 11 and 12 show the Macro-F1 scores and Micro-F1 scores of multi-label node classification for IMDB dataset.

As can be seen from Tables 5 and 6, among all algorithms, LINE has the worst performance while HIN_DRL performs the best. On the one hand, the average Macro-F1 score of HIN_DRL is 1.13% (relatively 4.9%) higher than that of Metapath2vec++, 1.41% (relatively 6.2%) higher than that of Metapath2vec, 1.96% (relatively 8.8%) higher than that of Node2vec, 2.24% (relatively 10.2%) higher than that of DeepWalk, and 10.53%(relatively 76.7%) higher than that of LINE. On the other hand, the average Micro-F1 score of HIN_DRL is 1.83% higher than that of Metapath2vec++ (relatively 4.1%), 2.00% higher than that of Metapath2vec (relatively 4.5%), 3.03% higher than that of DeepWalk (relatively 7.0%), 3.07% higher than that of Node2vec (relatively 7.1%), and 8.36% higher than that of LINE (relatively 22.1%).

It can also be seen by comparing Figs. 11 & 12 with Figs. 9 & 10 that, the improvement of HIN_DRL to the baseline methods in

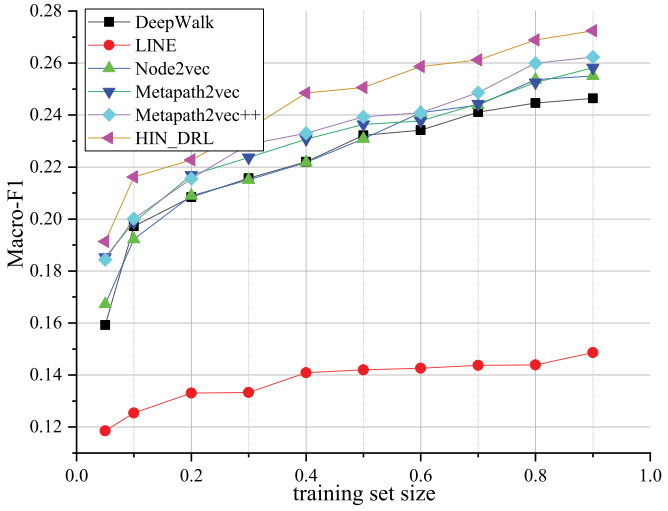



**Fig. 11.** Multi-label node classification of movie nodes in IMDB dataset (Macro-F1).

IMDB dataset is larger than that in Microsoft Academic dataset. The results also prove that the auxiliary storyline and release time information in movie network can be further utilized to improve the representation effect of movie nodes, thus benefit the multi-label node classification task on IMDB dataset.

Tables 7 & 8 and Figs. 13 & 14 show the Macro-F1 scores and Micro-F1 scores of multi-label node classification on YELP dataset. As expected, HIN_DRL also achieves the best results. Specifically, on one hand, the average Macro-F1 score of HIN_DRL is higher than that of Metapath2vec++ by 0.13% (relatively 15.7%), higher than that of Metapath2vec by 0.15% (relatively 18.5%), higher than that of Node2vec by 0.26% (relatively 37.1%), higher than that of DeepWalk by 0.30% (relatively 45.5%) and higher than that of LINE by 0.58% (relatively 152.6%). On the other hand, the average Micro-

**Table 7**
Multi-label node classification of business nodes in YELP dataset (Macro-F1).

| Training set size | 5% | 10% | 20% | 30% | 40% | 50% | 60% | 70% | 80% | 90% | Mean Macro-F1 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| DeepWalk | 0.004 | 0.004 | 0.005 | 0.005 | 0.005 | 0.007 | 0.007 | 0.009 | 0.009 | 0.012 | 0.007 |
| LINE | 0.003 | 0.003 | 0.003 | 0.003 | 0.003 | 0.004 | 0.004 | 0.004 | 0.005 | 0.007 | 0.004 |
| Node2vec | 0.004 | 0.004 | 0.005 | 0.005 | 0.006 | 0.008 | 0.009 | 0.009 | 0.010 | 0.012 | 0.007 |
| Metapath2vec | 0.004 | 0.004 | 0.005 | 0.005 | 0.0073 | 0.008 | 0.009 | 0.012 | 0.013 | 0.014 | 0.008 |
| Metapath2vec++ | 0.005 | 0.005 | 0.005 | 0.006 | 0.007 | 0.007 | 0.0092 | 0.012 | 0.013 | 0.015 | 0.008 |
| HIN_DRL | **0.006** | **0.006** | **0.006** | **0.007** | **0.008** | **0.008** | **0.010** | **0.013** | **0.015** | **0.018** | **0.010** |

**Table 8**
Multi-label node classification of business nodes in YELP dataset (Micro-F1).

| Training set size | 5% | 10% | 20% | 30% | 40% | 50% | 60% | 70% | 80% | 90% | Mean Micro-F1 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| DeepWalk | 0.117 | 0.118 | 0.127 | 0.129 | 0.140 | 0.145 | 0.148 | 0.149 | 0.158 | 0.171 | 0.140 |
| LINE | 0.117 | 0.120 | 0.131 | 0.133 | 0.134 | 0.143 | 0.144 | 0.152 | 0.154 | 0.155 | 0.139 |
| Node2vec | 0.117 | 0.122 | 0.127 | 0.129 | 0.148 | 0.149 | 0.149 | 0.150 | 0.152 | 0.164 | 0.141 |
| Metapath2vec | 0.137 | 0.139 | 0.144 | 0.146 | 0.147 | 0.148 | 0.151 | 0.170 | 0.176 | 0.196 | 0.155 |
| Metapath2vec++ | 0.146 | 0.147 | 0.147 | 0.148 | 0.156 | 0.163 | 0.169 | 0.171 | 0.174 | 0.1802 | 0.160 |
| HIN_DRL | **0.160** | **0.160** | **0.160** | **0.162** | **0.164** | **0.167** | **0.170** | **0.174** | **0.184** | **0.19** | **0.170** |

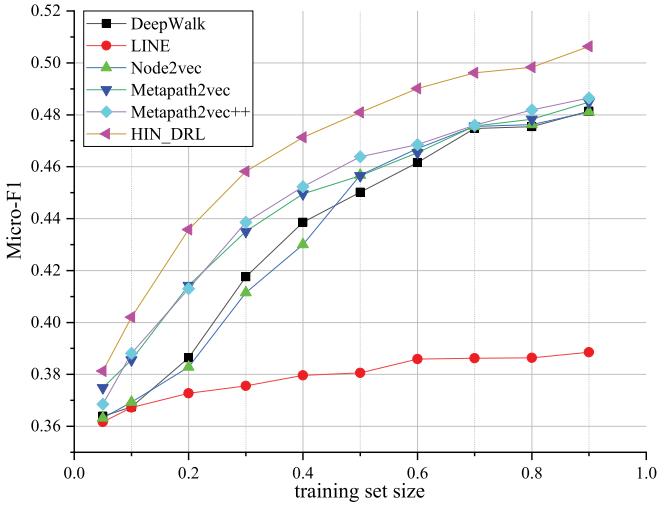



**Fig. 12.** Multi-label node classification of movie nodes in IMDB dataset (Micro-F1).

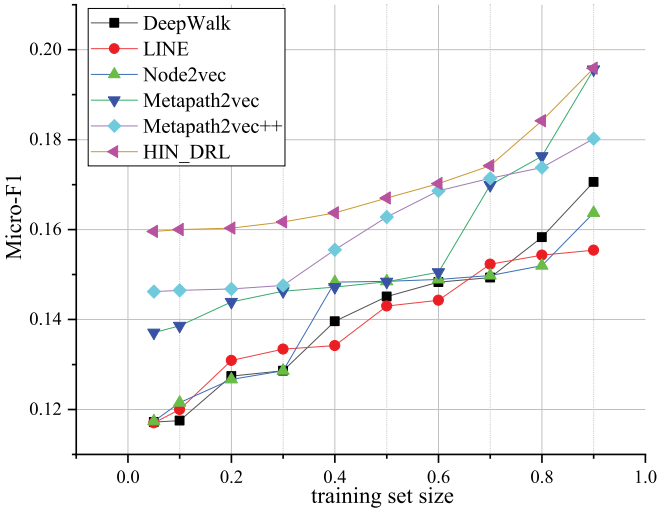



**Fig. 14.** Multi-label node classification of business nodes in YELP dataset (Micro-F1).

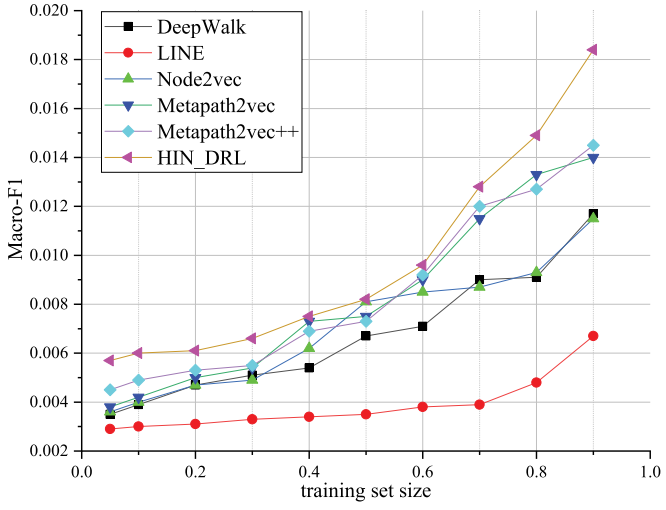



**Fig. 13.** Multi-label node classification of business nodes in YELP dataset (Macro-F1).

F1 of HIN_DRL is 0.97% higher than that of Metapath2ve++c (relatively 6.1%), 1.43% higher than that of Metapath2vec (relatively 9.2%), 2.92% higher than that of Node2vec (relatively 20.8%), 2.95% higher than that of DeepWalk (relatively 21.0%), and 3.12% higher than that of LINE (relatively 22.5%). In summary, the auxiliary short-text and post time information of YELP network can be further utilized to improve the effect of the multi-label node classification task on YELP dataset.

It can also be seen from Figs. 11 to 14 that, compared to the results of multi-class node classification in Figs. 9 & 10, the Macro-F1 scores and Micro-F1 scores of multi-label node classification of all algorithms are lower. And as the size of training set increases, the growth of the metrics is slower. This is because that multi-label node classification needs to correctly classify each label of network nodes, which makes it more challenge than multi-class node classification. Further compare the results in Figs. 11 & 12 with that in Figs. 13 & 14, we can see that the Macro-F1 and Micro-F1 scores of YELP dataset are much lower than those of IMDB dataset, which is because the categories of businesses are more blurred while those of movies are more differentiated. For example, a hotel can be tagged as a restaurant owing to its popular food or a gallery for a temporary special exhibition in YELP dataset.

**Table 9**
Node clustering results in Microsoft Academic dataset, IMDB dataset and YELP dataset. (NMI).

| Node type | Venues | Movies | Business |
|---|---|---|---|
| DeepWalk | 0.387 | 0.059 | 0.005 |
| LINE | 0.279 | 0.058 | 0.003 |
| Node2vec | 0.582 | 0.059 | 0.004 |
| Metapath2vec | 0.611 | 0.077 | 0.005 |
| Metapath2vec++ | 0.642 | **0.080** | 0.005 |
| HIN_DRL | **0.676** | 0.077 | **0.006** |

*5.2.2. Node clustering*

In this section, we verify the effect of HIN_DRL on node clustering. Same as the node classification experiments, the label information used by YELP and IMDB dataset comes from the source dataset, while the label information for Microsoft Academic dataset is manually supplemented from Google Scholar. Here we also use the entire dataset to learn the representation of network nodes first and then the low-dimensional representation is input into a clustering model. According to different datasets, FCM (Bezdek, 1981) or k-means is used for verification respectively, and NMI (Normalized Mutual Information) (Strehl & Ghosh, 2002) is used as the evaluation metric. Each experiment is repeated 10 times and the average NMI is reported.

Table 9 shows the clustering results of the comparison algorithms and HIN_DRL under different datasets.

It can be seen from Table 9 that, when clustering venue nodes, HIN_DRL performs the best, as the average NMI is about 3.36% higher than that of Metapath2vec++ (relatively 5.2%), 6.47% higher than that of Metapath2vec (relatively 10.6%), 9.34% higher than that of Node2vec (relatively 16.0%), 28.83% higher than that of DeepWalk (relatively 74.4%) and 39.65% higher than that of LINE (relatively 142.1%).

When clustering movie nodes, Metapath2vec++ performs the best, as the average value of NMI is about 0.27% higher than that of Metapath2vec and HIN_DRL (relatively 3.4%), 2.00% higher than that of DeepWalk (relatively 33.7%), 2.03% higher than that of Node2vec (relatively 34.4%) and 2.10% higher than that of LINE (relatively 36.0%).

When clustering business nodes, HIN_DRL also performs the best, as the average value of NMI is about 0.09% higher than that of Metapath2vec (relatively 18.6%), 0.10% higher than that of Metapath2vec++ (relatively 20.8%), 0.13% higher than that of DeepWalk (relatively 28.3%), 0.22% higher than that of Node2vec (relatively 61.5%) and 0.27% higher than that of LINE (relatively 88.5%).

Similar to the node classification tasks, the results obtained by the heterogeneous movie network and heterogeneous social network (learnt by FCM) are generally lower, while the results obtained by the heterogeneous academic network (learnt by k-means) are generally higher. And the heterogeneous algorithms (Metapath2vec, Metapath2vec++, HIN_DRL) always perform better than those homogeneous algorithms (DeepWalk, Node2vec, LINE). In addition, the results also demonstrate that it is meaningful that performing network representation learning by extracting the semantic information of networks, and maximizing the utilization of network information is also effective for learning the low-dimensional representation of nodes.

*5.2.3. Case study: visualization*

In this experiment, t-SNE algorithm (Maaten & Hinton, 2008) is used to visualize node representations in 2 dimensions, where the parameter for the confusion degree is 5, the learning rate is 2000, and the number of iterations is 2000. The labels of nodes are the fields of journals[6], and are marked with different colors in Fig. 15. Fig. 15 shows the 2-D graph of venue nodes representation learned in Microsoft Academic dataset by six different NRL algorithms, in which the closer the nodes with the same color are, the better result the algorithm has.

It can be seen that compared with other algorithms, HIN_DRL performs the best since it can effectively distinguish four classes. The effect of LINE is the worst, as it is impossible to gather any of the 10 categories. Metapath2vec, Metapath2vec++ and Node2vec can distinguish two classes, and DeepWalk can only distinguish one class, which is basically consistent with the case of node classification and node clustering tasks.

It can be further seen that the venue nodes marked with yellow diamond (label 5) is more easily to be distinguished by most algorithms, which corresponding to the field of Computer Vision & Pattern Recognition. In fact, the corresponding venues are indeed more distinguishable from other types of venues. The venue nodes marked with magenta plus sign (label 6) is difficult to be recognized, which is related to the field of Computer Systems, the corresponding journals and conferences are indeed difficultly divided from other types of journal conferences in actual situations.

**Table 10**
NMI, Macro-F1 and Micro-F1 of venue nodes in different timestamps.

| year | Mean Macro-F1 | Mean Micro-F1 | NMI |
|---|---|---|---|
| 2004 | 0.490 | 0.548 | 0.662 |
| 2010 | **0.509** | **0.610** | **0.676** |

*5.2.4. Case study: dynamics of node representation*

Here we analyze the dynamics of node representation by visualizing their 2 dimensional graph under two different timestamps of years 2004 and 2010.

In Table 10, we list the results of multi-class node classification and node clustering. Compare with those in year 2004, we get a better classification result as the Macro-F1 score is about 0.02% higher (relatively 3.9%), Macro-F1 score is about 0.06% higher (relatively 11.3%) and get a better clustering result as NMI is about 0.01% higher (relatively 2.1%) in year 2010.

In addition, as shown in Fig. 16, we can see that the relations between venues that belong to one field has changed over time. Some fields become tighter, such as Artificial Intelligence (label 1) and Computer Vision & Pattern Recognition (label 5), and some become looser, such as Computer Graphics (label 3). Besides, the 2-D visualization in year 2010 comes out a clearer layout, as more nodes with the same color cluster with each other than those in year 2004.

The above results show that, the more network information we have, the better result we get. And our method HIN_DRL can learn the different node representation under different timestamps, which thus can be used to analyze the dynamics of networks.

*5.2.5. Results discussion*

Based on the above experimental results on several typical machine learning tasks, including multi-class node classification, multi-label node classification and node clustering, we can summarize the following conclusions.

Firstly, for all the tasks on three different datasets, HIN_DRL always achieves the best effects in term of Macro-F1, Micro-F1 and

[6] 1. Artificial Intelligence, 2. Computational Linguistics, 3. Computer Graphics, 4. Computer Networks & Wireless Communication, 5. Computer Vision & Pattern Recognition, 6. Computing Systems, 7. Data Mining & Analysis, 8. Databases & Information Systems, 9. Human Computer Interaction, and 10. Theoretical Computer Science.

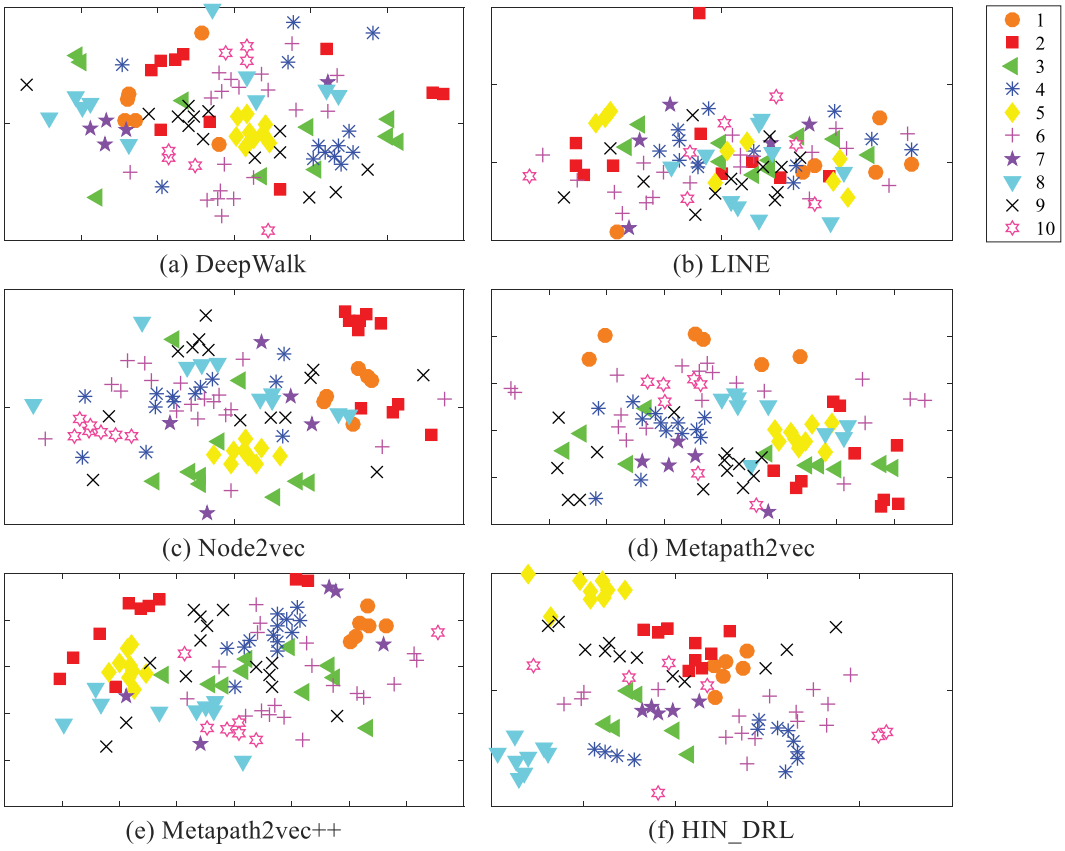



**Fig. 15.** 2-D visualization of venue node representations in Microsoft Academic dataset.

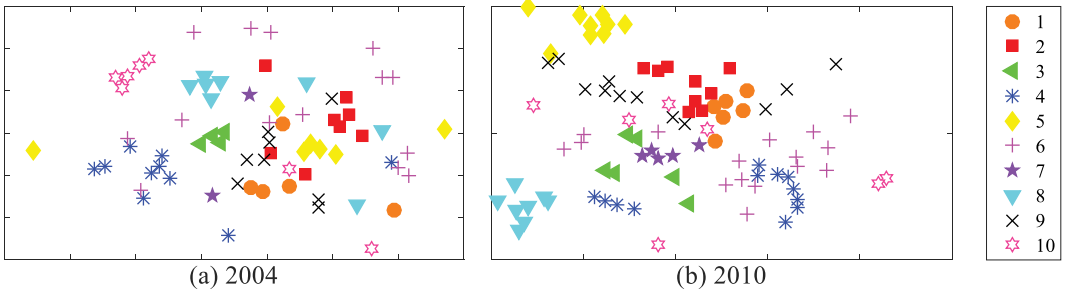



**Fig. 16.** 2-D visualization of venue nodes under different timestamps.

NMI, except for node clustering on IMDB dataset, which is next to Metapath2vec++. This proves that considering the meta-path based semantic information and the corresponding node sequences optimization strategies are effective for learning node representation in heterogeneous information network.

Secondly, on different dataset, the value ranges of Macro-F1, Micro-F1 and NMI for all algorithms are different. The results on Microsoft Academic dataset are the highest, those on IMDB dataset are generally lower, and those on YELP dataset are the lowest. Meanwhile, the improvement on different datasets of HIN_DRL to the baseline methods are different. The improvements of HIN_DRL on IMDB dataset is larger than that in Microsoft Academic dataset and YELP dataset. The first reason may be that multi-label node classification on IMDB is more challenging than multi-class node classification on Microsoft Academic dataset, and the categories of businesses are more blurred in YELP dataset. The second reason may be that the rich semantic information in IMDB can be effectively utilized by HIN_DRL.

Thirdly, the results on the case studies of visualization and dynamics further demonstrate that HIN_DRL can learn network representation dynamically and more effectively. We believe if combined with dynamic visualization technology, our theoretical method can be made as a real-time monitoring system where the rules of evolving networks would be easier found.

## 6. Conclusions

In this paper, a new NRL method based on edge-based random walk called HIN_DRL is proposed for learning the dynamic representation of heterogeneous information networks at different timestamps by following aspects of improving. First, it extracts and extends initial meta-paths, so as to provide richer semantic guidance for random walk in heterogeneous networks. Second, an edge-based dynamic random walk method called EDRW is adopted, where edge based transition probability matrix is constructed using network structure, semantics, text information and timestamps. Finally, two optimization strategies are used to dynamically adjust the length and number of node sequences during random walk process. Thus HIN_DRL can flexibility learn the dy-

namic and optimized node representation in evolving information network.

The proposed HIN_DRL can be adapted to many different kinds of heterogeneous information networks and perform network representation learning for multiple types of nodes at different timestamps. Empirically, we evaluate the quality of the latent representations learned by different methods over several classical heterogeneous network mining tasks in three real-world datasets. The results show that HIN_DRL can well learn the low-dimensional node representation and effectively improve the performance of downstream machine learning tasks. It is worth noting, unlike other approaches, HIN_DRL can not only learn the representation of heterogeneous networks by using semantics information based on meta-paths, but also do that for dynamic networks with timestamps.

However, there are still some limitations in our method. First of all, HIN_DRL adopts an edge-based dynamic random walk to generate node sequences, and the edge-based transition probability is calculated using topic distributions of nodes and timestamp distance, therefore it cannot be applied to those networks without text and timestamp information. Secondly, all the meta-paths extracted via the automate method (AutoMP) are equally used to guide the random walk while ignoring their weights, which may affect the effect of representation learning.

Future work involves difficulties and challenges left in this area. For example, how to learn a real-time representation of a network more efficiently will be one of the interesting work, and incremental based approach may be competitive. We also believe that it would be an interesting research direction that a more general similarity measure for edges is studied as we utilize the topic-based similarity between three pairs of nodes, where the text information of networks is indispensable. Besides, with the extensive research and application of graph neural network, it may be a good direction in network representation. Finally, it would be meaningful to build a real-time system based on dynamic visualization to monitor evolving networks.

## Declaration of Competing Interest

We wish to confirm that there are no known conflict of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome. We confirm that the manuscript has been read and approved by all named authors and that there are no other persons who satisfied the criteria for authorship but are not listed. We further confirm that the order of authors listed in the manuscript has been approved by all of us. We confirm that we have given due consideration to the protection of intellectual property associated with this work and that there are no impediments to publication, including the timing of publication, with respect to institutions concerning intellectual property. We understand that the Corresponding Author is the soe contact for the Editorial process (including Editorial Manager and direct communication with the office). He/she is responsible for communicating with the other authors about progress, submissions of revisions and final approval of proofs. We confirm that we have provided a current, correct email address which is accessible by the Corresponding Author and which has been configured to accept email from mllu@bupt.edu.cn .

## Credit authorship contribution statement

**LU Meilian:** Supervision, Methodology, Writing - original draft, Writing - review & editing. **YE Danna:** Investigation, Methodology, Writing - original draft, Validation.

## Acknowledgments

This work was supported in part by National Natural Science Foundation of China under Grant 61602048 and the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, China.

## References

Anderson, W. N., & Morley, T. D. (1985). Eigenvalues of the Laplacian of a graph. *Linear and Multilinear Algebra, 18*(2), 141–145.
Cao, S., Lu, W., & Xu, Q. (2015). GraRep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM international conference on information and knowledge management, CIKM'15*.
Chen, W., Liu, C., Yin, J., Yan, H., & Zhang, Y. (2017). Mining E-commercial data: A text-rich heterogeneous network embedding approach. In *Proceedings of the IEEE international joint conference on neural networks, IJCNN*.
Dai, H., Wang, Y., Trivedi, R., & Song, L. (2017, August). Deep coevolutionary network: Embedding user and item features for recommendation. Proceedings of ACM SIGKDD international conference on knowledge discovery and data mining, KDD'17.
Dong, Y., Chawla, N. V., & Swami, A. (2017). Metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23th ACM SIGKDD international conference on knowledge discovery and data mining, KDD'17*.
Du, L., Wang, Y., Song, G., Lu, Z., & Wang, J. (2018). Dynamic network embedding: an extended approach for skip-gram based network embedding. In *Proceedings of the 27th International Joint Conferences on Artificial Intelligence, IJCAI'18*.
Fu, T., Lee, W., & Lei, Z. (2017). HIN2Vec: Explore Meta-paths in Heterogeneous Information Networks for Representation Learning. In *Proceedings of the 26th ACM International Conference on Information and Knowledge Management, CIKM'17*.
Goyal, P., & Ferrara, E. (2018). Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems, 151*, 78–94.
Grover, A., & Leskovec, J. (2016). Node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, KDD'16*.
Gui, H., Liu, J., Tao, F., Jiang, M., Norick, B., Kaplan, L., & Han, J. (2017). Embedding learning with events in heterogeneous information networks. *IEEE Transactions on Knowledge and Data Engineering, 29*(11), 2428–2441.
Gupta, M., Kumar, P., & Bhasker, B. (2017). HeteClass: A meta-path based framework for transductive classification of objects in heterogeneous information networks. *Expert Systems with Applications, 68*, 106–122.
Hamilton, W. L., Ying, R., & Leskovec, J. (2017, December). Inductive Representation Learning on Large Graphs. Advances in neural information processing systems, NIPS'17.
Huang, Z., & Mamoulis, N. (2017, January). Heterogeneous Information Network Embedding for Meta Path based Proximity. arXiv preprint. arXiv:1701.05291v1.
Bezdek, James C. (1981). *Pattern recognition with fuzzy objective function algorithms*. Boston: Springer (Chapter 4).
Keikha, M. M., Rahgozar, M., & Asadpour, M. (2018). Community aware random walk for network embedding. *Knowledge-Based Systems, 148*, 47–54.
Li, J., Dani, H., Hu, X., Tang, J., Chang, Y., & Liu, H. (2017). Attributed network embedding for learning in a dynamic environment. In *Proceedings of the 26th ACM international conference on information and knowledge management, CIKM'17*.
Ma, G., Lu, C. T., He, L., Yu, P. S., & Ragin, A. B. (2017). Multi-view Graph Embedding with Hub Detection for Brain Network Analysis. In *Proceedings of the 17th IEEE international conference on data mining, ICDM'17*.
Ma, J., Cui, P., & Zhu, W. (2018). DepthLGP: learning embeddings of out-of-sample nodes in dynamic networks. In *Proceedings of the 32nd AAAI conference on artificial intelligence, AAAI'18*.
Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research, 9*, 2579–2605.
Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. Cambridge: Cambridge University Press (Chapter 14).
Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed Representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 3111–3119.
Nguyen, G. H., Lee, J. B., Rossi, R. A., Ahmed, N. K., Koh, E., & Kim, S. (2018). Continuous-time dynamic network embeddings. In *Proceedings of the 27th international conference on world wide web conference, WWW'18*.
Pan, S., Wu, J., Zhu, X., Zhang, C., & Wang, Y. (2016). Tri-party deep network representation. In *Proceedings of the 25th international joint conference on art artificial intelligence*. IJCAI'16.
Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). DeepWalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining*. KDD'14.
Perozzi, B., Kulkarni, V., & Skiena, S. (2016, May). Walklets: Multiscale Graph Embeddings for Interpretable Network Classification. arXiv preprint. arXiv:1605.02115.
Pham, P., & Do, P. (2019). W-MetaPath2VecProceedings of the topic-driven meta–path-based model for large-scaled content-based heterogeneous information network representation learning. *Expert Systems with Applications, 123*, 328–344.
Qiao, L., Zhao, H., Huang, X., Li, K., & Chen, E. (2019). A structure-enriched neural network for network embedding. *Expert Systems with Applications, 117*, 300–311.

Qu, M., Tang, J., Shang, J., Ren, X., Zhang, M., & Han, J. (2017). An attention-based collaboration framework for multi-view network representation learning. In *Proceedings of the 26th ACM international conference on information and knowledge management*. CIKM'17.

Ribeiro, L. F., Saverese, P. H., & Figueiredo, D. R. (2017). Struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. KDD'17.

Roweis, S. T., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science, 290*(5500), 2323–2326.

Strehl, A., & Ghosh, J. (2002). Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research, 3*, 583–617.

Sun, Y., Han, J., Zhao, P., Yin, Z., Cheng, H., & Wu, T. (2009). Rankclus: Integrating clustering with ranking for heterogeneous information network analysis. In *Proceedings of the 12th international conference on extending database technology: advances in database technology*. EDBT 2009.

Sun, Y., Han, J., Yan, X., Yu, P. S., & Wu, T. (2011). Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. In *Proceedings of the 37th international conference on very large data bases*. VLDB'11.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., & Mei, Q. (2015a). LINE: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web conference, WWW'15*.

Tang, J., Qu, M., & Mei, Q. (2015b). PTE: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 21st ACM SIGKDD international conference on knowledge discovery and data mining, KDD'15*.

Tu, C., Zhang, Z., Liu, Z., & Sun, M. (2017). TransNet: Translation-based network representation learning for social relation extraction. In *Proceedings of the 26th international joint conference on artificial intelligence, IJCAI'17*.

Wang, D., Cui, P., & Zhu, W. (2016). Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, KDD'16*.

Xu, L., Wei, X., Cao, J., & Yu, P. S. (2017a). Embedding of Embedding (EOE): Joint Embedding for Coupled Heterogeneous Networks. In *Proceedings of the 10th ACM international conference on web search and data mining, WSDM'17*.

Xu, L., Wei, X., Cao, J., & Yu, P. S. (2017b). Embedding identity and interest for social networks. In *Proceedings of the 25th international conference on world wide web companion, WWW'17*.

Yang, C., Sun, M., Liu, Z., & Tu, C. (2017). Fast network embedding enhancement via high order proximity approximation. In *Proceedings of the 26th international joint conference on artificial intelligence, IJCAI'17*.

Yu, Y., Lu, Z., Liu, J., Zhao, G., Wen, J. R., & Zheng, K. (2017). RUM: Network representation learning through multi-level structural information preservation. arXiv preprint. arXiv:1710.02836.

Yu, W., Cheng, W., Aggarwal, C. C., Zhang, K., Chen, H., & Wang, W. (2018). NetWalk: A Flexible Deep Embedding Approach for Anomaly Detection in Dynamic Networks. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery and data mining, KDD'18*.

Zhou, L., Yang, Y., Ren, X., Wu, F., & Zhuang, Y. (2018). Dynamic network embedding by modeling triadic closure process. In *Proceedings of the 32nd AAAI conference on Artificial Intelligence, AAAI'18*.

Zhu, D., Cui, P., Zhang, Z., Pei, J., & Zhu, W. (2018). High-order proximity preserved embedding for dynamic networks. *IEEE Transactions on Knowledge and Data Engineering, 30*(11), 2134–2144.

Zuo, Y., Liu, G., Lin, H., Guo, J., Hu, X., & Wu, J. (2018). Embedding temporal network via neighborhood formation. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery and data mining KDD'18*.