

Tree Structure-Aware Graph Representation Learning via Integrated Hierarchical Aggregation and Relational Metric Learning

Ziyue Qiao^{1,2}, Pengyang Wang³, Yanjie Fu³, Yi Du^{1,*}, Pengfei Wang⁴, Yuanchun Zhou¹

¹Computer Network Information Center, Chinese Academy of Sciences, Beijing

²University of Chinese Academy of Sciences, Beijing

³Department of Computer Science, University of Central Florida, Orlando

⁴Alibaba DAMO Academy, Alibaba Group, China

qiaoziyue@cnic.cn, {pengyang.wang@knights., yanjie.fu@}ucf.edu, {duyi, zyc}@cnic.cn

Abstract—While Graph Neural Network (GNN) has shown superiority in learning node representations of homogeneous graphs, leveraging GNN on heterogeneous graphs remains a challenging problem. The dominating reason is that GNN learns node representations by aggregating neighbors' information regardless of node types. Some work is proposed to alleviate such issue by exploiting relations or meta-path to sample neighbors with distinct categories, then use attention mechanism to learn different importance for different categories. However, one limitation is that the learned representations for different types of nodes should own different feature spaces, while all the above work still project node representations into one feature space. Moreover, after exploring massive heterogeneous graphs, we identify a fact that multiple nodes with the same type always connect to a node with another type, which reveals the many-to-one schema, *a.k.a.* the hierarchical tree structure. But all the above work cannot preserve such tree structure, since the exact multi-hop path correlation from neighbors to the target node would be erased through aggregation. Therefore, to overcome the limitations of the literature, we propose T-GNN, a tree structure-aware graph neural network model for graph representation learning. Specifically, the proposed T-GNN consists of two modules: (1) the integrated hierarchical aggregation module and (2) the relational metric learning module. The integrated hierarchical aggregation module aims to preserve the tree structure by combining GNN with gated recurrent unit to integrate the hierarchical and sequential neighborhood information on the tree structure to node representations. The relational metric learning module aims to preserve the heterogeneity by embedding each type of nodes into a type-specific space with distinct distribution based on similarity metrics. In this way, our proposed T-GNN is capable of simultaneously preserving the heterogeneity and the tree structure inherent in heterogeneous graphs. Finally, we conduct extensive experiments to show the outstanding performance of T-GNN in tasks of node clustering and classification, inductive node clustering and classification, and link prediction.

Index Terms—Graph Neural Network; Graph Representation learning; Metric Learning; Heterogeneous Graph.

I. INTRODUCTION

Graph Neural Network (GNN) is a family of graph representation learning approaches that encode node features into low-dimensional representation vectors by aggregating neighbors'

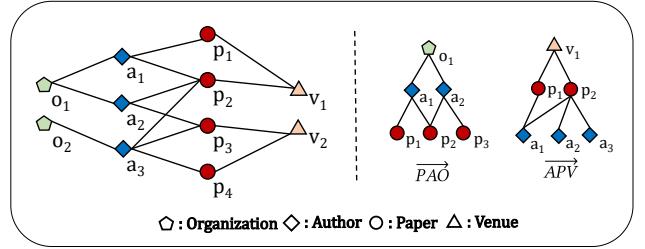


Fig. 1: An illustration of hierarchical tree structures in academic heterogeneous graph.

information [1]. GNN has drawn increasing attention in recent decades, due to the superior performance in tremendous real-world applications, such as recommender systems [2], [3], urban computing [4], [5], chemistry [6], etc.

While GNN models conform well with the learning tasks on homogeneous graphs, leveraging GNN on learning node representations for heterogeneous graphs remains a challenging problem. First of all, heterogeneous graphs consist of nodes with different types, but vanilla GNN indiscriminately aggregates neighbors' information regardless of node types. Second, the relations between different types of nodes usually reveal the structure of many-to-one, *a.k.a.* the hierarchical tree structure, which indicates the schema that multiple nodes with the same type connect to a node with another type. For example, Figure 1 shows an instance in the academia graph, papers p_1 , p_2 and p_3 are linked to author a_1 and a_2 by the relation $\text{Paper} \xrightarrow{\text{authored}} \text{Author}$ (abbreviated as \overrightarrow{PA}), also authors a_1 and a_2 are linked to organization o_1 by the relation $\text{Author} \xrightarrow{\text{employed}} \text{Organization}$ (abbreviated as \overrightarrow{AO}). By composing \overrightarrow{PA} and \overrightarrow{AO} , the nodes of type P, A and O form a three-level hierarchical tree with structure \overrightarrow{PAO} and o_1 as the root node, this tree implies a two-hop structured neighborhood for o_1 , where the papers are in the ground level and the authors are in the second level.

In literature, some work are proposed to alleviate such issue. The main idea is sampling neighbors with distinct categories via different types of relations or meta-path (*i.e.*, composed relations), and use attention mechanism or coefficients to assign different importance to different categories of nodes

*Corresponding author.

The research is supported by the Natural Science Foundation of China under Grant No. 61836013, the National Key Research and Development Plan of China (No. 2016YFB0501901), Ministry of Science and Technology Innovation Methods Special work Project under grant 2019IM020100, Beijing Nova Program of Science and Technology under Grant No. Z191100001119090.

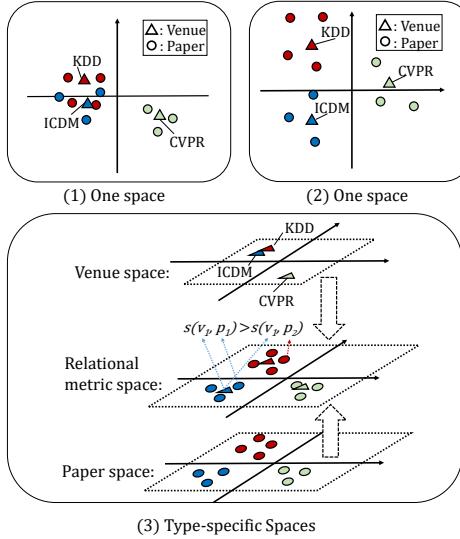


Fig. 2: An illustration of respectively embedding three venues and their corresponding papers into same spaces and type-specific spaces. (1)(2) shows if different types of nodes are embedded into one space. (3) shows by embedding nodes into type-specific spaces.

as aggregating neighbors [7]–[11]. However, there still exists limitations. First, some methods [11]–[14] only differentiate neighbor information by atomic relations, i.e., the one-hop links, and propagate information of multi-hop neighborhoods by stacking multiple layers. However, the exact multi-hop path correlation from neighbors to the target node would be erased through aggregation. Second, for some methods [2], [9] using meta-path to sample neighbors in multi-hop, given the target node, meta-paths directly transfer the information between the two end nodes of the meta paths, while ignore the relay nodes in the middle. On the other hand, meta-path based neighborhoods preserve less structural information relatively than the tree-based neighborhoods. Third, the learned representations for different types of nodes should own different feature spaces, while all the above work still projects node representations into one feature space.

Therefore, to overcome the limitations of the literature, we propose a Tree structure-aware Graph Neural Network(T-GNN) for heterogeneous graph representation learning. Next, we outline the key steps and insights of our proposed T-GNN.

Preserving Tree Structure via Integrated Hierarchical Aggregation. As mentioned above, multi-hop neighborhoods in heterogeneous graphs can be decomposed to different trees, which can denote different organizational forms of neighbors and relations. Also on these trees, the neighbor information with different distance can be further formulated as a sequence with variable length from the bottom level to the top level. This indicates that we can sequentially aggregate the information of lower level nodes and propagate it to the higher level nodes to finally contribute to the root nodes' representations. Therefore, we propose an integrated hierarchical aggregation module to preserve tree structures of heterogeneous graphs. Specifically, we differentiate tree structured neighborhoods by introducing

hierarchical tree schema, and partition neighborhoods into organized trees with different categories. Then we combine GNN models with Gated Recurrent Unit (GRU) modules to aggregate hierarchical and sequential neighborhood information on these trees to node representations.

Preserving Heterogeneity via Relational Metric Learning. Most heterogeneous graph representation learning methods aim to embed different types of nodes into the same feature space. However, in one space, different types of nodes share the same polynomial distribution but the similarity between nodes of different types could be incompatible and dampen with each other.

For example in Figure 2(a), the closeness of two similar venues would results in the papers published by these two venues tend to be inseparable, while in Figure 2(b), well separated papers makes their two venues distant and the proximity of them would not be well preserved. Such incompatibility may also exists between other types of nodes in one embedding space. Model optimization usually need to learn specific distributions of nodes for certain tasks. Therefore, we propose a relational metric learning module to embed each type of nodes into a type-specific space with independent distribution. Specifically, we introduce metrics to measure the similarities between nodes with different types, as shown in Figure 2(c), the similarity between nodes with same type is calculated in their type space, and that between nodes with different types is calculated in a relation type-specific metric space. Finally, we leverage a relation-aware graph context loss based on random walk and negative sampling to train the whole model.

In summary, we propose a T-GNN model to improve the capability of GNN on learning node representations of heterogeneous graphs. The proposed T-GNN includes two key modules, (1) hierarchical aggregation module, which aims to preserve the tree structure by integrating GNN with GRU; (2) relational metric learning module, which aims to preserve the heterogeneity by mapping different type of nodes into different embedding space with incorporating similarity measurement. The main contributions of our work are summarized as follow.

- 1) To our best knowledge, we are the first to introduce the tree structures inherent in heterogenous graphs into the architecture of graph neural network model to learn node representations.
- 2) We propose a tree structure-aware graph neural network model named T-GNN, which contains aggregation modules and sequence propagation modules to capture both the node attributes and the information in multi-hop neighborhoods with different hierarchical organizations into node representations.
- 3) We propose a relational metric learning module for unsupervised graph representation learning that embed nodes into type-specific spaces with independent distributions.
- 4) We conduct extensive experiments to evaluate the performance of our proposed model on three datasets. The results demonstrate the superiority of our proposed method over several state-of-the-art methods.

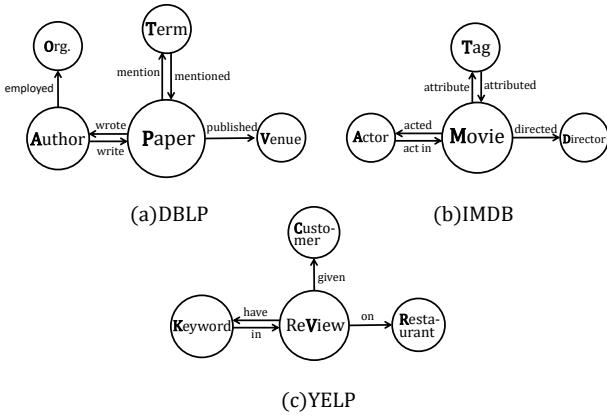


Fig. 3: Schemas of three heterogeneous graphs. (a) DBLP Academic graph. (b) IMDB Movie graph. (c) YELP review graph. Each solid line arrows represent a relation type, each circles represent one node type and the bold letters represent their abbreviations.

II. PRELIMINARIES

In this section, we give formal definitions of some related concepts and notations.

Definition 1 (Heterogeneous Graph). A heterogeneous graph is denoted as $G = (N, E, \mathcal{T}, \mathcal{R})$, in which each node $n \in N$ is associated with a node type mapping functions $\phi(n) : N \rightarrow \mathcal{T}$ and each edge $e \in E$ is associated with a relation type mapping function $\phi(e) : E \rightarrow \mathcal{R}$. This suggests that G has multiple node types and relation types, and $|\mathcal{T}| + |\mathcal{R}| > 2$.

Figure 3 shows the schemas of three heterogeneous graphs. We resolve the undirected links between nodes into fine-grained directed relations to present the hierarchical structures. The directions of relations are consistent with the structures of many-to-one between different types of nodes.

Definition 2 (Meta-path). In a heterogeneous graph $G = (N, E, \mathcal{T}, \mathcal{R})$, a meta-path is defined as a path in the form of $t_0 \xrightarrow{r_1} t_1 \xrightarrow{r_2} \dots \xrightarrow{r_l} t_l$ (abbreviated as $t_0t_1\dots t_l$), where $r_i \in \mathcal{R}$ and $t_i \in \mathcal{T}$, the sequence of relations represents the composite relation $r = r_1 \circ r_2 \circ \dots \circ r_l$ between the node type t_0 and t_l .

For example, we can design PAP , APV as meta-paths, PAP means two papers are coauthored by a same person, APV means an author publish a paper on a venue. Meta-paths are widely utilized to extract paths in previous works. Similarly, in our paper, we define hierarchical tree schemas to extract hierarchical trees.

Definition 3 (Hierarchical Tree Schema). A Hierarchical Tree Schema s is a directed meta-path in the form of $s = \{t_0 \xrightarrow{r_1} t_1 \xrightarrow{r_2} \dots \xrightarrow{r_m} t_m\}$ (abbreviated as $t_0t_1\dots t_m$), where the direction of each r_i is from t_{i-1} to t_i . These schemas are used to extract trees where t_m is the type of root node and each node in the i th level has the type t_i , representing the $(m-i)$ -hop neighbors of the root node.

Definition 4 (Hierarchical Tree Structured Neighborhood). Given a hierarchical tree schema $s = \{t_0 \xrightarrow{r_1} t_1 \xrightarrow{r_2} \dots \xrightarrow{r_m}$

$t_m\}$ and a node n_o with type t_m , we first sample the neighbors of n_o connecting by relation r_m to the $(m-1)$ th level, then we sample neighbors of nodes in the $(m-1)$ th level connecting by relation r_{m-1} to the $(m-2)$ th level, and so on, sample nodes until to the 0th level. Finally, we can obtain a hierarchical tree with n_o in the level m as the root node, and the nodes in level $m-k$ with type t_{m-k} is n_o 's k -hop neighbors.

For example, in academic dataset, the hierarchical tree schema \overrightarrow{TPV} represent a kind of structured neighborhood for venue nodes, where venue is the root nodes on the 2-th level, its papers are on the 1-th level, and the terms of these papers are on the 0-th level. \overrightarrow{AMD} in movie dataset represents a structured neighborhood for each director, containing its movies on the 1-th level and the actors of these movies on the 0-th level.

Given a node n_o with node type $t_m = \phi(n_o)$, we can extract k kinds of hierarchical tree schemas ended with n_o 's type: $\mathcal{S}_t = \{s_1, s_2, \dots, s_k\}$, that is, $\forall s \in \mathcal{S}_t, s = \{t_0 \xrightarrow{r_1} t_1 \xrightarrow{r_2} \dots \xrightarrow{r_m} t_m\}$. To differentiate the neighborhoods that different schemas refer to, we define each scheme in \mathcal{S}_t is not a subsequence of another, i.e., $\forall s_i, s_j \in \mathcal{S}_t \wedge i \neq j \rightarrow s_i \not\subseteq s_j$.

These hierarchical tree schemas partition n_o 's neighborhoods into different categories of hierarchical trees. These hierarchical trees preserve different semantics and sequence information and are discriminated in aggregation model. For example, \overrightarrow{TPV} and \overrightarrow{APV} represent two kinds of structured neighborhood of venues. For \overrightarrow{TPV} , first term information is aggregated to papers, and then papers which contain the terms' information are aggregated to the venues, while for \overrightarrow{APV} , the aggregation order is authors, papers and venues.

Definition 5 (Heterogeneous Graph Representation Learning). Given a heterogeneous graph $G = (N, E, \mathcal{T}, \mathcal{R})$ learning the representation is to learn node type-specific mapping functions $\{f_t : N_t \rightarrow \mathbb{R}_t^d\}, t \in \mathcal{T}$, which projects the nodes in N_t (node sets of type t) to a representation vector in the d' -dimensional latent space \mathbb{R}_t^d , corresponding to the feature space of the node type t .

III. PROPOSED MODEL

In this section, we introduce the T-GNN model for heterogeneous graph representation learning. Specifically, we combine the gated recurrent neural networks and GNN to process the information on hierarchical tree structured neighborhoods. Then we use attention mechanism to measure the importance of the information aggregated on different trees and integrate them into node representations. Finally, we present a relational metric learning module for model optimization.

A. Hierarchical Aggregation

For a node n_o , we have discussed we can use hierarchical tree schemas to divide its multi-hop neighborhood into multiple structured trees, each of those represents a distinctive neighborhood for n_o . Given the hierarchical tree extracted by one schema s where n_o is the root node, and the initial feature

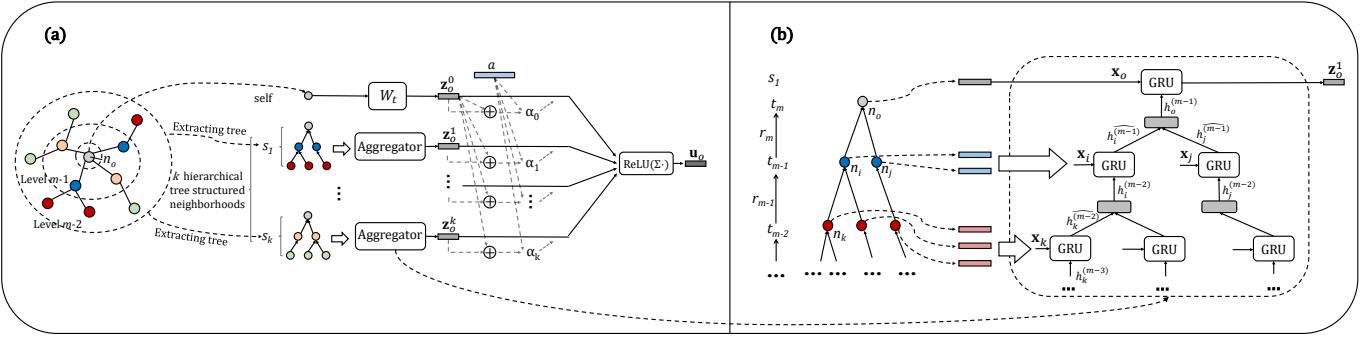


Fig. 4: An illustration of our proposed propagation model. (a) The whole propagation model which consists of hierarchical neighborhood extracting, hierarchical aggregating and representation integrating. (b) The architecture of the proposed hierarchical aggregation.

vectors $\mathbf{x}_i \in \mathbb{R}^{d_t}$ (d_t : the initial feature vector dimension) of each node on this tree, our intuition is to aggregate the information from all the neighbors under n_o 's level on the tree as well as \mathbf{x}_o to form its representations.

As the neighbor information in the propagation can be regarded as sequential inputs divided by levels and GNN can not process sequential data. We conduct **Gated Recurrent Unit** (GRU) module combining with GNN on this tree to encode \mathbf{x}_o and its neighbors information into a schema-specific output \mathbf{z}_o^s , the basic recurrence of the aggregation modules and propagation modules on hierarchical trees is:

$$\begin{aligned} \widehat{h_i^{(0)}} &= \mathbf{x}_i, \quad \phi(n_i) = t_0 \\ \widehat{h_i^{(a)}} &= \text{GRU}(\mathbf{x}_i, h_i^{(a-1)}), \quad \phi(n_i) = t_a \\ h_i^{(a-1)} &= \text{AGGREGATE}_{r_a}(\{\widehat{h_j^{(a-1)}}, n_j \in N_i^{r_a}\}) \end{aligned} \quad (1)$$

where $0 < a \leq m$, $\widehat{h_i^{(a)}}$ represents the output hidden state of node n_i which is on the a th level of the tree with schema s , i.e., . The hidden state $h_i^{(a-1)}$ represents the neighborhood message of n_i passing from all its neighbors on the $(a-1)$ th level. The $\text{GRU}(\mathbf{x}_i, h_i^{(a-1)})$ is formulated as:

$$\begin{aligned} z_t &= \sigma(A_z \mathbf{x}_i + B_z h_i^{(a-1)}) \\ r_t &= \sigma(A_r \mathbf{x}_i + B_r h_i^{(a-1)}) \\ \tilde{h}_i^{(a)} &= \tanh[A_h \mathbf{x}_i + B_h(r_t \circ h_i^{(a-1)})] \\ \widehat{h_i^{(a)}} &= z_t \circ \tilde{h}_i^{(a)} + (1 - z_t) \circ \widehat{h}_i^{(a)} \end{aligned} \quad (2)$$

Where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function and \circ is element-wise multiplication. The aggregator function $\text{AGGREGATE}_{r_a}(\cdot)$ is a information aggregator of n_i 's sub-level neighbors specific for relation type r_a , which could be a mean pooling layer, max pooling layer, or a GNN aggregator, such as mean aggregator, LSTM aggregator, pooling aggregator introduced in GraphSAGE [15], graph attentional layer introduced in GAT [16]. In our paper, we use weighted mean aggregator, then the $h_i^{(a-1)}$ in Eq.1 is reformulated as:

$$h_i^{(a-1)} = c_{ij}^{r_a} \sum_{n_j \in N_i^{r_a}} W_{r_a} \widehat{h_j^{(a-1)}} \quad (3)$$

where r_a is the relation type between level a and level $a-1$ on s that connect n_i with its sub-level neighbors, N_i^r denotes the set of neighbors indices of node n_i under the relation $r \in \mathcal{R}$. W_r is a trainable weight matrix specific for relation type r . c_{ij}^r is the weight or normalization constant of the edge (n_i, n_j) with type r .

After the m levels' propagation on hierarchical trees structured neighborhood, the hidden output of node n_o on schema s can be obtained by:

$$\mathbf{z}_o = \text{GRU}(\mathbf{x}_o, h_o^{(m-1)}) \quad (4)$$

Where $\mathbf{z}_o \in \mathbb{R}^{d'} (d': \text{representation dimension})$. To make model tuning easy, we also set the dimension of hidden states as d' . In the neighbor information aggregation, parameters in GRU modules are shared and those in GNN modules are shared for same relation types. In practice, to save the calculation time, we use tensor operation and avoid the repeated aggregating on different trees with the same schemas, for example, for schema \overrightarrow{TP} of paper nodes and \overrightarrow{TPA} of author nodes, the aggregating processes from T to P are same for two schema and we only calculate once.

We have noticed that the gated neural units have been applied to some graph neural network models [17]–[19], majority of which are originated from GGNN [20]. Their main idea is stacking k -layer GNNs, and using GRU module to process the k hidden layer outputs of each node as the sequenced information. Our proposed model is different from this idea: 1) We use the GRU module to help GNN directly aggregates the sequence information composed of neighbor nodes with arbitrary length, 2) Instead of stacking a fixed number of layers, our proposed T-GNN can stack different numbers of propagation layers for different node types according to their neighborhood depths.

B. Neighborhood Information Integrating

Then given the hierarchical tree schema set $S_t = \{s_1, s_2, \dots, s_k\}$ of n_o , we can obtain the hidden output set $\mathbf{Z}_o = \{\mathbf{z}_o^1, \mathbf{z}_o^2, \dots, \mathbf{z}_o^k\}$ of n_o and $\mathbf{z}_o^i \in \mathbb{R}^{d'}$. These hidden representations that contain different neighborhood information may make different contributions to n_o 's final representation. We employ the attention mechanism to combine these k

hidden representations with n_o 's feature vectors into the final representation vector of n_o , formulated as:

$$\mathbf{u}_o = \text{ReLU} \left(\alpha_o^* \cdot W_t \mathbf{x}_o + \sum_{\mathbf{z}_o^i \in \mathbf{Z}_o} \alpha_o^i \cdot \mathbf{z}_o^i \right) \quad (5)$$

Where $\mathbf{u}_o \in \mathbb{R}^{d'}$ is the final representation vector of n_o , α_o^* indicates the importance of different hidden representations for n_o , $W_t \in \mathbb{R}^{d' \times d}$ is a type-specific linear transformation to project the features vectors of nodes with type t into a hidden representation vector, which represent the feature information aggregated from each node itself. For brevity, we denote that $\mathbf{z}_o^0 = W_t \mathbf{x}_o$ and add it into representation set \mathbf{Z}_o , then \mathbf{u}_o can be reformulated as:

$$\mathbf{u}_o = \text{ReLU} \left(\sum_{\mathbf{z}_o^i \in \mathbf{Z}_o} \alpha_o^i \cdot \mathbf{z}_o^i \right) \quad (6)$$

We leverage self-attention to learn the attention weights α_o^i , which are calculated via softmax function:

$$\alpha_o^i = \frac{\exp\{\text{LeakyReLU}(a^T[\mathbf{z}_o^0 || \mathbf{z}_o^i])\}}{\sum_{\mathbf{z}_o^j \in \mathbf{Z}_o} \exp\{\text{LeakyReLU}(a^T[\mathbf{z}_o^0 || \mathbf{z}_o^j])\}} \quad (7)$$

which can be interpreted as the importance of the hidden representation \mathbf{z}_o^i , the higher the α_o^i , the more important \mathbf{z}_o^i . LeakyReLU denotes the leaky version of a Rectified Linear Unit, $a \in \mathbb{R}^{1 \times 2d'}$ is the attention parameter.

Then we can apply the final node representation vector \mathbf{u}_o to the loss functions in supervised tasks, semi-supervised tasks, or unsupervised tasks, and optimize the propagation model via back propagation.

C. Optimization via Relational Metric learning

To perform heterogeneous graph representation learning, in this section, we leverage a graph context loss to optimize the model. Given a heterogeneous graph $G = (N, E, \mathcal{T}, \mathcal{R})$, we aim to learn effective node representations by maximizing the probability of any node n_i having its neighbor node n_c :

$$\arg \max_{\theta} \sum_{n_i \in N} \sum_{r \in \mathcal{R}} \sum_{n_c \in N_r(n_i)} \log p(n_c | n_i, \theta) \quad (8)$$

where $N_r(n_i)$ is a set of neighbors of n_i , and $\forall n_c \in N_r(n_i)$ connects n_i with relation r , θ represents the parameters of the whole model. $p(n_c | n_i, \theta)$ is the probability of any node n_i having its neighbor node n_c , defined as a softmax function:

$$p(n_c | n_i, \theta) = \frac{\exp(s(\mathbf{u}_i, \mathbf{u}_c))}{\sum_{n_j \in N_t} \exp(s(\mathbf{u}_i, \mathbf{u}_j))} \quad (9)$$

where \mathbf{u}_i represents the encoded representation vector of n_i , $s(\mathbf{u}_i, \mathbf{u}_j)$ is the similarity between node n_i and n_j . N_t represents the node set of type t and $t = \phi(n_c)$. The softmax function is normalized with respect to the node type of the context n_c , so each type of the neighborhood in the output layer is specified to one distinct multinomial distribution.

As we aim to embed each type $t \in \mathcal{T}$ of nodes into a distinct space, we set $s(\mathbf{u}_i, \mathbf{u}_j) = \mathbf{u}_i^T \mathbf{u}_j$ if n_i and n_j have same type and in the same embedding space. For each node pair $(\mathbf{u}_i, \mathbf{u}_j)$ with distinct node type t_a and t_b and connected by a relation with type r (which can be an atomic relation or a composite relation), we formulate the similarity $s(\mathbf{u}_i, \mathbf{u}_j)$ by introduce two candidate similarity metrics specific for r , which are described as:

- **Bilinear.** A multiplicative similarity metric. We introduce a bilinear metric $M_r \in \mathbb{R}^{d' \times d'}$, which is shared by each node pair $(\mathbf{u}_i, \mathbf{u}_j)$ with relation type r and specific to different relation types to avoid the similarity calculation on respective semantics to dampen each other.

$$s(\mathbf{u}_i, \mathbf{u}_j) = \mathbf{u}_i^T M_r \mathbf{u}_j \quad (10)$$

- **Perceptron.** A additive similarity metric. We first input \mathbf{u}_i and \mathbf{u}_j to node-type-specific multilayer perceptrons with output metric dimension d_m , then add the outputs of them, next with a tanh activation layer we can obtain the hidden representation vector of node pair $(\mathbf{u}_i, \mathbf{u}_j)$. Then a relation-type-specific metric $m_r \in \mathbb{R}^{d_m}$ is multiplied to calculate the similarity.

$$s(\mathbf{u}_i, \mathbf{u}_j) = m_r^T \tanh(M_{t_a} \mathbf{u}_i + M_{t_b} \mathbf{u}_j) \quad (11)$$

The metrics are trainable parameters and are learned to measure which \mathbf{u}_j under the relation r is closer to \mathbf{u}_i , whose mechanism is similar to the mechanism of attention models. Note that the parameters of metrics are shared for same relation type r , while different relation types make use of different metrics and the corresponding distance on these relations of nodes is calculated in different metric spaces,

Then, we adopt the popular negative sampling method proposed in [21] to sample negative nodes to increase the optimization efficiency. Then, the probability $\log p(n_c | n_i, \theta)$ can be approximately defined as:

$$\log \sigma(s(\mathbf{u}_i, \mathbf{u}_c)) + \sum_{n_j \in N_{neg}^i} \log \sigma(-s(\mathbf{u}_i, \mathbf{u}_j)) \quad (12)$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function and $N_{neg}^i \subseteq N_t$ is a negative node set for n_i sampled from a pre-computed node frequency distribution $P_t(n_i)$, we set $P(n_i) \propto f_{n_i}^{3/4}$, where $f_{n_i}^{3/4}$ is the frequency of n_i in \mathcal{P} . Then, we conduct random walks on the G to sample a set of paths \mathcal{P} . Therefore, we can use skip-gram model on these paths and reformulate the objective in eq.8 as follows:

$$\mathcal{L} = - \sum_{p \in \mathcal{P}} \sum_{n_i \in p} \sum_{n_c \in C_i^k} \log p(n_c | n_i, \theta) + \lambda \|\theta\|^2 \quad (13)$$

where k is the windows size of context, C_i^k is the context set containing the previous k context nodes and next k context nodes of n_i in the path p . Parameter λ controls penalty of regularization for over-fitting. Finally, we use a mini-batch Adam optimizer to minimize \mathcal{L} and optimize the parameters in the whole model.

TABLE I: Statistics of three datasets

Datasets	Node (# Number)	Hierarchical Tree Schema
DBLP	Paper #20,552	$\overrightarrow{AP}, \overrightarrow{TP}$
	Author #19,247	\overrightarrow{TPA}
	Venue #12	$\overrightarrow{TPV}, \overrightarrow{APV}$
IMDB	Movie #3,630	$\overrightarrow{AM}, \overrightarrow{TM}$
	Actor #13,156	\overrightarrow{TMA}
	Director #1,932	$\overrightarrow{AMD}, \overrightarrow{TMD}$
YELP	Costumer #2,536	\overrightarrow{KVC}
	Restaurant #3,332	\overrightarrow{KVR}

IV. EXPERIMENT

In this section, We conduct several experiments to validate the performance of our proposed T-GNN model on three datasets. The results demonstrate the superiority of our proposed method over several state-of-the-art methods. All codes of our method are publicly available¹.

A. Datasets

To evaluate the proposed method, we use three kinds of real-world datasets: academic graphs (DBLP), movie graphs (MOVIE), and review graph (YELP). The detailed descriptions of these datasets are as follows:

- **DBLP²:** DBLP is an academic bibliography with millions of publications. We extract a subset of DBLP with four areas: *Data Mining (DM)*, *Database (DB)*, *Natural Language Processing (NLP)* and *Computer Vision (CV)* from the year of 2013 to 2017. For each area, we choose three top venues³ and related papers(P), terms(T), authors(A) to construct an heterogeneous graph.
- **IMDB⁴:** IMDB dataset contains knowledge about movies. We extract three genres of movie information from IMDB: *Action*, *Comedy* and *Drama*. Then we construct a movie heterogeneous graph which contains movies(M), actors(A), directors(D) and Tags(T).
- **YELP⁵:** YELP is a datasets containing the data of restaurants reviews. We extract a subset of YELP containing costumers(C), review(V), keywords(K) and restaurants(R) from the information of restaurants with types: *American*, *Chinese*, *Japanese* and *French* to form a restaurant review heterogeneous graph.

We select some types of nodes and use graph representation learning methods to learn their representation vectors, Table I shows the statistics of these nodes and their corresponding hierarchical tree schemas used in our proposed model.

¹<https://github.com/joe817/T-GNN>

²<https://www.aminer.cn/citation>

³DM: KDD, ICDM, WSDM. DB: SIGMOD, VLDB, ICDE. NLP: ACL, EMNLP, NAACL. CV: CVPR, ICCV, ECCV.

⁴<https://www.kaggle.com/tmdb/tmdb-movie-metadata>

⁵<https://www.kaggle.com/yelp-dataset/yelp-dataset>

B. Comparison Methods

To validate the performance of our proposed method, we compare with some state-of-art baselines, including homogeneous heterogeneous graph representation learning method (DeepWalk), heterogeneous graph representation learning methods (Metapath2vec, RHINE), homogeneous graph neural networks (GraphSAGE, GAT) and heterogeneous graph neural networks (HAN, HetGNN). The brief descriptions of these baselines are as follows:

- **DeepWalk:** DeepWalk [22] is a graph representation learning method based on random walks and skip-gram model to learn latent node representations.
- **Metapath2vec:** Metapath2vec [23] uses meta-path based random walks to construct the heterogeneous neighborhood of nodes and then leverages a heterogeneous skip-gram model to perform node representations. We leverage the meta-paths APVPA, AMDMA and CRC in DBLP, IMDB, and Yelp respectively.
- **RHINE:** RHINE [24] explore the different structural characteristics of relations in HINs and present two structure-related measures for two distinct categories of relations: IR and AR. We select the same relations in their paper on DBLP and YELP, for IMDB, we select AM, AMD as IR, and TM, TMD, TMA as AR.
- **GraphSAGE:** GraphSAGE [15] learn node representations through different aggregation function form local neighborhood of nodes. We use the GCN module as the aggregator of GraphSAGE.
- **GAT:** GAT [16] consider the attention mechanism and measure impacts of different neighbors feature information by multi-head self-attention neural network.
- **HAN:** HAN [9] leverages node level attention and semantic-level attention to respectively learn the importance of neighbors based on same meta-path and different meta-paths. We select meta-paths CRC, RCR for YELP, and meta-paths same as theirs for DBLP, IMDB.
- **HetGNN:** HetGNN [11] jointly considered heterogeneous neighbors sampling, node heterogeneous contents encoding, type based neighbors aggregation, and heterogeneous types combination.

We use Par2vec [25] to pre-train the text content of nodes and set the dimension of initial feature vector as 128. For the proposed T-GNN, we set all the hidden layer dimensions and final representation dimension of nodes as 128. We use the Adam optimizer with a learning rate of 10^{-3} and the l2 penalty parameter λ is 10^{-4} . We set the windows size k of skip-gram model as 2, the size of negative samples as 3 for all datasets, the similarity metric is chosen to Perceptron and the metric space dimension d_m is 32. As GAT and HAN only mention semi-supervised objective function for node representations, for a fair comparison, we optimize their models same as our method to learn unsupervised node representations. For the baselines, we set the dimension of nodes in three datasets also as 128 and the other hyper-parameter setting are based on default values or the values specified in their own papers. All

the experiments are repeated many times to make sure the results can reflect the performances of methods.

TABLE II: Results of Clustering

Dataset	DBLP		IMDB		YELP	
	Metric	NMI	ARI	NMI	ARI	NMI
DeepWalk	0.735	0.616	0.023	0.015	0.260	0.279
Metapath2vec	0.864	0.899	0.096	0.091	0.261	0.282
RHINE	0.866	0.902	0.055	0.036	0.342	0.351
GraphSAGE	0.865	0.912	0.128	0.135	0.396	0.433
GAT	0.855	0.897	0.114	0.113	0.413	0.457
HAN	0.900	0.933	0.136	0.144	0.413	0.458
HetGNN	0.891	0.939	0.131	0.139	0.403	0.440
T-GNN	0.916	0.955	0.145	0.152	0.420	0.484

TABLE III: Results of Multi-class Classification

Dataset	DBLP		IMDB		YELP	
	Metric(F1)	Micro	Macro	Micro	Macro	Micro
DeepWalk	0.905	0.896	0.478	0.473	0.703	0.660
Metapath2vec	0.922	0.918	0.505	0.509	0.705	0.660
RHINE	0.927	0.920	0.449	0.448	0.726	0.676
GraphSAGE	0.962	0.943	0.583	0.584	0.739	0.748
GAT	0.967	0.958	0.543	0.542	0.744	0.758
HAN	0.979	0.971	0.596	0.598	0.746	0.759
HetGNN	0.983	0.979	0.594	0.593	0.730	0.753
T-GNN	0.997	0.996	0.608	0.609	0.760	0.772

C. Clustering and Classification

First, we conduct clustering and classification tasks to evaluate the performance of the methods. For DBLP, we label papers by areas of their venues and label authors by their representative areas (the area with the majority of their papers). For IMDB, we label the movies by genres and label actors by the genre with the majority of their acted movies. For YELP, we label the restaurant by type. Then we evaluate these nodes' representations by clustering and multi-class classification. We learn the node representations on the heterogeneous graphs by each model. For clustering task, we feed node representations into the K-Means algorithm, the number of clusters is set to the true number of classes for each dataset, we evaluate the clustering performance in terms of normalized mutual information (NMI) and adjusted rand index (ARI). For multi-class classification task, the learned node representations are used as the input to a logistic regression classifier, the proportion of training and valid data is set to 50%, 10%, the remaining nodes are used for test. We use both Micro-F1 and Macro-F1 as classification evaluation metrics.

Results. Table II shows the performance evaluation of clustering on three datasets and Table III shows the classification results. The proposed T-GNN model consistently outperform all baselines in terms of two tasks, showing that it can learn more effective node representations than other methods. The relative improvements (%) of T-GNN over the best baselines range from 1.7%-10.7% for clustering task and 1.4%-2.2% for classification task. We can also observe 1) GNN based graph representation learning models achieve better

performance than the traditional graph representation learning methods on these two tasks, that is because besides the graph structure information, GNNs can also capture node attributes information to node representations. 2) Heterogeneous models perform better than homogeneous models because they can differentiate different types of relations and capture more semantic information to node representations. 3) T-GNN's performance is better than two heterogeneous graph neural network models HAN and HetGNN, showing that it can better capture the information of heterogeneous neighborhood.

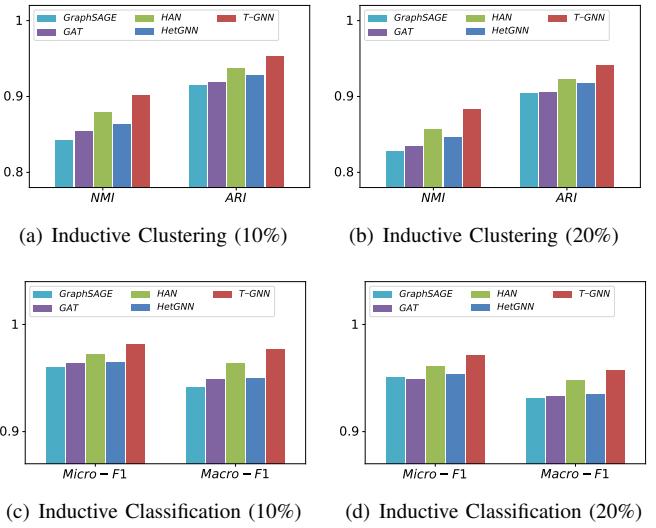


Fig. 5: Results of inductive clustering and inductive multi-class classification, where the percentages in captions represent the invisible ratios of nodes.

D. Inductive Clustering and Classification

In this task, we evaluate the inductive capability of our proposed T-GNN on DBLP dataset and compare with the graph neural network based baselines GraphSAGE, GAT, HAN and HetGNN. Specifically, for author nodes and paper nodes which to be evaluated, we randomly set a part of them as the invisible nodes for graph representation learning methods, and use the rest visible nodes and the links between them on the graph to train the models. Then we use the learned models on the whole graph to infer the representations of all invisible nodes. Finally, we use the inferred representations as the input of clustering tasks. For the classification tasks, the visible nodes are used to train the classifier, and the invisible nodes are used to evaluate. The labels and evaluation metrics are the same as the previous clustering and classification tasks. We report the results of the invisible nodes ratio in as 10% and 20% respectively.

Results. Figure 5 shows the performance of GNN models on the inductive learning task. Our proposed T-GNN model achieves the best performance among all the homogeneous and heterogeneous models. Specifically, for the comparison of heterogeneous models, our proposed T-GNN outperforms HAN and HetGNN, because the representations of invisible

nodes are mainly inferred by their neighborhood information, but HAN uses meta-path to sample neighbors, ignoring the intermediate nodes; HetGNN uses random walks to sample neighbors, them both unable to process tree structured information and path correlation on multi-hop neighborhood.

E. Link Prediction

In this section, we aim to evaluate the performance of our graph representation learning methods on the link prediction task, which is a practical problem in many applications such as user/item recommendation. We formulate this task as a binary classification problem to predict whether a relation link between two nodes exists. We consider two types of links: links of two author collaborating one paper in DBLP dataset, and links of a user giving a review to an restaurant in YELP dataset. Specifically, for DBLP, we sample the collaborating links before 2013 for training, in 2013 for validation and after 2013 for test. For YELP, we randomly sample 60% reviewing links for training, 10% for validation, and the rest for testing. In training, we remove the links in the validation and testing sets and use graph representation learning methods on graphs to learn the node representations. We use the element-wise multiplication of two candidate nodes' representations as the link representation, then input link representations into a binary logistic classifier. Also negative links (not connected in graphs) with three times the number of true links are randomly sampled to the training, validation and testing sets. We use AUC and F1 as evaluate metrics.

Results. Table IV shows the prediction results on two datasets. We can find that our proposed T-GNN model achieves the best performance or is comparable to the best methods on the link prediction task. We believe the reason is that our proposed T-GNN model has outstanding ability in preserve heterogeneity into node representations, which helps to overcome the mutual interference of different relations on the node distributions, and improve the quality of node representations.

TABLE IV: Results of Link Prediction

Dataset	DBLP (Collaborating)		YELP (Reviewing)	
	AUC	F1	AUC	F1
DeepWalk	0.691	0.539	0.519	0.392
Metapath2vec	0.723	0.622	0.518	0.390
RHINE	0.729	0.630	0.552	0.410
GraphSAGE	0.713	0.584	0.665	0.536
GAT	0.719	0.603	0.652	0.550
HAN	0.743	0.635	0.655	0.551
HetGNN	0.760	0.660	0.646	0.536
T-GNN	0.824	0.764	0.663	0.571

F. Comparison of Variant Models

In order to further verify the effectiveness of the metrics on learning node representations, we design three variant heterogeneous graph representation learning models based on T-GNN and different output models: T-GNN_{dp}, T-GNN_{pe}, T-GNN_{bi}. T-GNN_{dp} do not use metrics and leverage dot

product as the similarity $s(n_i, n_j)$ of any two nodes n_i and n_j , T-GNN_{pe} use the Perceptron as the similarity metric of nodes with different types and T-GNN_{bi} use Bilinear, both of them are introduced in Section III-C. We evaluate these three models on the node clustering task and classification task. The experiment setting are same as mentioned above. The results are shown in Figure 6. It is evident that using two kinds of metrics achieves better performance. Besides, the performance of T-GNN_{pe} is better than T-GNN_{bi}. We believe the reason is that there are more parameters in Perceptron metric than Bilinear metric, leading to Perceptron metric is better in preserve the complicated proximities between multiple nodes.

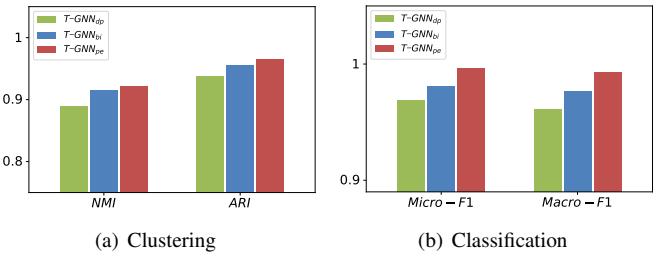


Fig. 6: Performance Evaluation of Variant Models.

G. Visualization

We have discussed that our model can embed each type of nodes into type-specific space with distinct distribution. In this section, we compare our model with metapath2vec and GraphSAGE by visualizing node representations with different types. Specifically, we first learn the node representations in the year of 2015 of four venue: CVPR, ACL, KDD, SIGMOD in DBLP data, representing four research areas. Then we use t-SNE to respectively project author representations and paper representations into 2-dimensional spaces.

The results are shown in Figure 7. We can observe that while metapath2vec cluster papers and authors into small groups, some nodes belonging to the same research area are far away from each other, and some belonging to different areas are mixed with each other, GraphSAGE can roughly partition nodes with different research areas, but the boundaries between different clusters are not sharp, and many nodes overlap at the same points. We can observe that in the visualization of T-GNN, 1) the representations of paper nodes in the same area cluster closely and can be well distinguished from each other. Author nodes are also embedded well comparing with other two methods, but some of them are embedded on the boundary between different clusters, which is because some authors may have multiple research areas. 2) Authors and papers in their own space have independent distributions, showing high intra-class similarities and distinct boundaries between different research areas.

V. RELATED WORK

Graph Representation Learning Graph representation learning, also known as network embedding, has become one of the most popular research interests in data mining

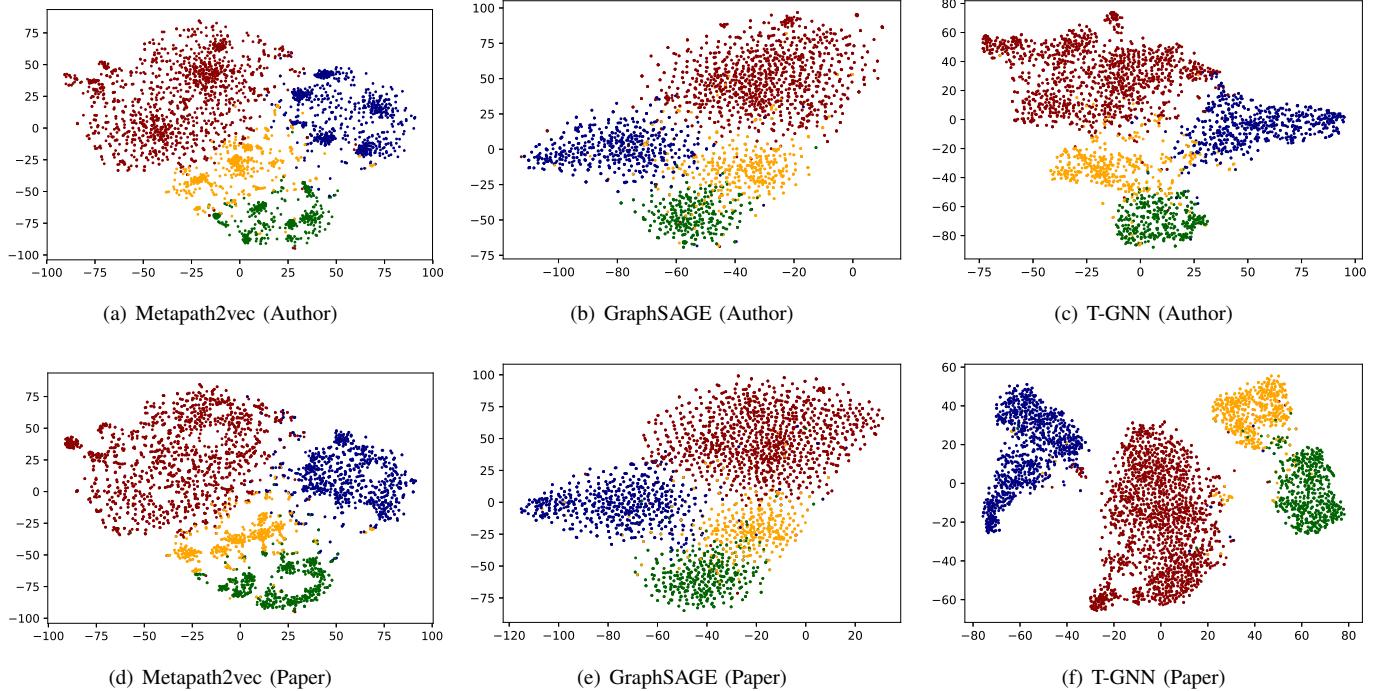


Fig. 7: t-SNE Visualization of representation distribution of authors and papers learned by three different graph representation learning methods: Metapath2vec, GraphSAGE and T-GNN. (a), (b), (c) denotes the author representations learned by these three methods and (d), (e), (f) denotes the paper representations. The four different research areas which authors and papers belong to are colored differently.

recently. Network embedding method aim to embed network into lower dimensional space which preserve the network property and node proximity, the learned node embeddings can be utilized in many graph mining tasks. In homogeneous networks embedding methods, DeepWalk [22] and Node2Vec [26] use random walk strategy on network and skip-gram [27], [28] model to learn the representation of each node in network. LINE [29] aims to learn the node embedding that preserve both first-order and second-order proximities. Some use deep neural network for homogeneous network embedding, such as DNGR [30] and SDNE [31]. Different with homogeneous graph, heterogeneous information network contains multiple types of nodes and relations, many real world graph can be modeled as HINs. Metapath2Vec [23] designs a meta-path based random walk and utilizes skip-gram to perform heterogeneous graph embedding, HINE [32] define an objective function modeling the meta path based proximities in the embedded low-dimensional space. HIN2Vec [33] learns the embeddings of HINs by conducting multiple prediction training tasks jointly, RHINE [24] explore the relation characteristics in HINs and partition them into two categories, then use different models to optimize the node representations. Besides, Some methods introduce metrics learning into heterogeneous network embedding, HEER [34] embeds HINs via edge representations that are further coupled with properly-learned heterogeneous metrics, PME [35] models node and link heterogenities in elaborately designed relation-specific spaces.

Graph Neural Network Graph neural networks extend the deep neural network to encode the node features and local

neighborhood into low-dimensional representation vectors [1]. Kipf et al. propose GCN [36], which designs a graph convolutional network via a localized first-order approximation of spectral graph convolutions. GraphSAGE [15] propose an general aggregating strategy containing multiple aggregators, such as Mean aggregator, LSTM aggregator and Pooling aggregator. For heterogeneous graph, R-GCN [12] introduces a relational graph convolutional network to link prediction task and entity classification task. Zhang et al. propose a heterogeneous graph neural network model HetGNN [11] which considers both types and heterogeneous attributes of nodes. RSHN [13] utilize graph structure and implicit relation structural information to simultaneously learn node and edge type embedding. Some work introduce attention mechanisms into network architectures, GAT [16] employs self-attention mechanism to measure impacts of different neighbors and combine their impacts to obtain node representations. HAN [9] employs node-level and semantic-level attentions on heterogeneous network to learn the importance of neighbors based on meta-paths. Also some work introduce gated recurrent units, GGNN [20] aim learning representations of the internal state during the process of producing a sequence of outputs. Inspired by this idea, [17] proposed a novel encoder-decoder architecture for graph-to-sequence learning. [37] propose general conceptual message passing neural networks framework that subsumes most GNNs and evaluate it on molecular property prediction task. [38] propose a differentiable graph pooling module to learn hierarchical representations of graphs on task of graph classification.

VI. CONCLUSION REMARKS

In this paper, we study the problem of tree structure-aware graph representation learning. To effectively aggregate the multi-hop neighborhood, which usually present the structure of hierarchical trees in most heterogeneous graphs, we propose a tree structure-aware graph neural network model, which contains aggregation modules and sequence propagation modules to transform information of neighbors within multi-hop links to the root nodes. To overcome the problem of the mutual interference on the distributions of different types of nodes, we propose a relational metric learning module to optimize model, which utilize metrics to calculate node similarities on relation-specific metric spaces and embed nodes into type-specific spaces. From the experiment, the heterogeneous graph representation learning method we proposed can better extract the rich information of neighborhoods and improve the qualities of node representations.

REFERENCES

- [1] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, “Graph neural networks: A review of methods and applications,” *arXiv preprint arXiv:1812.08434*, 2018.
- [2] S. Fan, J. Zhu, X. Han, C. Shi, L. Hu, B. Ma, and Y. Li, “Metapath-guided heterogeneous graph neural network for intent recommendation,” in *Proceedings of the 25th ACM SIGKDD*, 2019, pp. 2478–2486.
- [3] L. Wu, Y. Yang, K. Zhang, R. Hong, Y. Fu, and M. Wang, “Joint item recommendation and attribute inference: An adaptive graph convolutional network approach,” *arXiv preprint arXiv:2005.12021*, 2020.
- [4] P. Wang, Y. Fu, H. Xiong, and X. Li, “Adversarial substructured representation learning for mobile user profiling,” in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 130–138.
- [5] P. Wang, Y. Fu, Y. Zhou, K. Liu, X. Li, and K. Hua, “Exploiting mutual information for substructure-aware graph representation learning,” in *Proceedings of the 29th International Joint Conference on Artificial Intelligence. AAAI Press*, 2020.
- [6] C. W. Coley, W. Jin, L. Rogers, T. F. Jamison, T. S. Jaakkola, W. H. Green, R. Barzilay, and K. F. Jensen, “A graph-convolutional neural network model for the prediction of chemical reactivity,” *Chemical science*, vol. 10, no. 2, pp. 370–377, 2019.
- [7] A. Sankar, X. Zhang, and K. C.-C. Chang, “Meta-gnn: metagraph neural network for semi-supervised learning in attributed heterogeneous information networks,” in *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 2019, pp. 137–144.
- [8] X. Fu, J. Zhang, Z. Meng, and I. King, “Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding,” in *Proceedings of The Web Conference 2020*, 2020, pp. 2331–2341.
- [9] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, “Heterogeneous graph attention network,” in *The World Wide Web Conference*, 2019, pp. 2022–2032.
- [10] Z. Qiao, Y. Du, Y. Fu, P. Wang, and Y. Zhou, “Unsupervised author disambiguation using heterogeneous graph convolutional network embedding,” in *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 2019, pp. 910–919.
- [11] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla, “Heterogeneous graph neural network,” in *Proceedings of the 25th ACM SIGKDD*, 2019, pp. 793–803.
- [12] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, “Modeling relational data with graph convolutional networks,” in *European Semantic Web Conference*. Springer, 2018, pp. 593–607.
- [13] S. Zhu, C. Zhou, S. Pan, X. Zhu, and B. Wang, “Relation structure-aware heterogeneous graph neural network,” in *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2019, pp. 1534–1539.
- [14] P. Wang, J. Gui, Z. Chen, J. Rhee, H. Chen, and Y. Fu, “A generic edge-empowered graph convolutional network via node-edge mutual enhancement,” in *Proceedings of The Web Conference 2020*, 2020, pp. 2144–2154.
- [15] W. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” in *Advances in Neural Information Processing Systems*, 2017, pp. 1024–1034.
- [16] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, “Graph attention networks,” *arXiv preprint arXiv:1710.10903*, 2017.
- [17] D. Beck, G. Haffari, and T. Cohn, “Graph-to-sequence learning using gated graph neural networks,” *arXiv preprint arXiv:1806.09835*, 2018.
- [18] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson, “Structured sequence modeling with graph convolutional recurrent networks,” in *International Conference on Neural Information Processing*. Springer, 2018, pp. 362–373.
- [19] X. Bresson and T. Laurent, “Residual gated graph convnets,” *arXiv preprint arXiv:1711.07553*, 2017.
- [20] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, “Gated graph sequence neural networks,” *arXiv preprint arXiv:1511.05493*, 2015.
- [21] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in *Advances in neural information processing systems*, 2013, pp. 3111–3119.
- [22] B. Perozzi, R. Al-Rfou, and S. Skiena, “Deepwalk: Online learning of social representations,” in *SIGKDD*. ACM, 2014, pp. 701–710.
- [23] Y. Dong, N. V. Chawla, and A. Swami, “metapath2vec: Scalable representation learning for heterogeneous networks,” in *SIGKDD*. ACM, 2017, pp. 135–144.
- [24] Y. Lu, C. Shi, L. Hu, and Z. Liu, “Relation structure-aware heterogeneous information network embedding,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 4456–4463.
- [25] Q. Le and T. Mikolov, “Distributed representations of sentences and documents,” in *ICML*, 2014, pp. 1188–1196.
- [26] A. Grover and J. Leskovec, “node2vec: Scalable feature learning for networks,” in *SIGKDD*. ACM, 2016, pp. 855–864.
- [27] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *arXiv preprint arXiv:1301.3781*, 2013.
- [28] X. Rong, “word2vec parameter learning explained,” *arXiv preprint arXiv:1411.2738*, 2014.
- [29] J. Tang, M. Qu, M. Wang, M. Zhang *et al.*, “Line: Large-scale information network embedding,” in *WWW. International World Wide Web Conferences Steering Committee*, 2015, pp. 1067–1077.
- [30] S. Cao, W. Lu, and Q. Xu, “Deep neural networks for learning graph representations,” in *AAAI*, 2016, pp. 1145–1152.
- [31] D. Wang, P. Cui, and W. Zhu, “Structural deep network embedding,” in *Proceedings of the 22nd ACM SIGKDD*, 2016, pp. 1225–1234.
- [32] Z. Huang and N. Mamoulis, “Heterogeneous information network embedding for meta path based proximity,” *arXiv preprint arXiv:1701.05291*, 2017.
- [33] T. Fu, W. Lee, and Z. Lei, “Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning,” in *CIKM*. ACM, 2017, pp. 1797–1806.
- [34] Y. Shi, Q. Zhu, F. Guo, C. Zhang, and J. Han, “Easing embedding learning by comprehensive transcription of heterogeneous information networks,” in *SIGKDD*, 2018, pp. 2190–2199.
- [35] H. Chen, H. Yin, W. Wang, H. Wang, Q. V. H. Nguyen, and X. Li, “Pme: projected metric embedding on heterogeneous networks for link prediction,” in *SIGKDD*, 2018, pp. 1177–1186.
- [36] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” *arXiv preprint arXiv:1609.02907*, 2016.
- [37] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, “Neural message passing for quantum chemistry,” in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR.org, 2017, pp. 1263–1272.
- [38] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, “Hierarchical graph representation learning with differentiable pooling,” in *Advances in neural information processing systems*, 2018, pp. 4800–4810.