MAGNN: Metapath Aggregated Graph Neural Network for Heterogeneous Graph Embedding

Xinyu Fu
The Chinese University of Hong Kong
Hong Kong, China
xyfu@cse.cuhk.edu.hk

Jiani Zhang
The Chinese University of Hong Kong
Hong Kong, China
jnzhang@cse.cuhk.edu.hk

Ziqiao Meng
The Chinese University of Hong Kong
Hong Kong, China
zqmeng@cse.cuhk.edu.hk

Irwin King
The Chinese University of Hong Kong
Hong Kong, China
king@cse.cuhk.edu.hk

ABSTRACT

A large number of real-world graphs or networks are inherently heterogeneous, involving a diversity of node types and relation types. Heterogeneous graph embedding is to embed rich structural and semantic information of a heterogeneous graph into low-dimensional node representations. Existing models usually define multiple metapaths in a heterogeneous graph to capture the composite relations and guide neighbor selection. However, these models either omit node content features, discard intermediate nodes along the metapath, or only consider one metapath. To address these three limitations, we propose a new model named *Metapath Aggregated Graph Neural Network* (MAGNN) to boost the final performance. Specifically, MAGNN employs three major components, i.e., the node content transformation to encapsulate input node attributes, the intra-metapath aggregation to incorporate intermediate semantic nodes, and the inter-metapath aggregation to combine messages from multiple metapaths. Extensive experiments on three real-world heterogeneous graph datasets for node classification, node clustering, and link prediction show that MAGNN achieves more accurate prediction results than state-of-the-art baselines.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; **Learning latent representations**; • **Information systems** → **Social networks**.

KEYWORDS

Heterogeneous graph; Graph neural network; Graph embedding

ACM Reference Format:
Xinyu Fu, Jiani Zhang, Ziqiao Meng, and Irwin King. 2020. MAGNN: Metapath Aggregated Graph Neural Network for Heterogeneous Graph Embedding. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3366423.3380297

WWW '20, April 20–24, 2020, Taipei, Taiwan

ACM ISBN 978-1-4503-7023-3/20/04.
https://doi.org/10.1145/3366423.3380297

1 INTRODUCTION

Many real-world datasets are naturally represented in a graph data structure, where objects and the relationships among them are embodied by nodes and edges, respectively. Examples include social networks [14, 29], physical systems [2, 10], traffic networks [18, 34], citation networks [1, 14, 16], recommender systems [26, 35], knowledge graphs [3, 24], and so on. The unique non-Euclidean nature of graphs renders them difficult to be modeled by traditional machine learning models. For the neighborhood set of each node, there is no order or size limit. However, most statistical models assume an ordered and fixed-size input lying in the Euclidean space. Therefore, it would be beneficial if nodes could be represented by meaningful low-dimensional vectors in the Euclidean space and then be taken as the input for other machine learning models.

Different graph embedding techniques have been proposed for the graph structure. LINE [25] generates node embeddings by exploiting the first-order and second-order proximity between nodes. Random-walk-based methods including DeepWalk [21], node2vec [13], and TADW [32] feed node sequences generated by random walks to a skip-gram model [19] to learn node embeddings. With the rapid development of deep learning, graph neural networks (GNNs) have been proposed, which learn the graph representations using specially designed neural layers. Spectral-based GNNs, including ChebNet [8] and GCN [16], perform graph convolution operations in the Fourier domain of an entire graph. Recent spatial-based GNNs, including GraphSAGE [14], GAT [28], and many other variants [17, 34, 35], address the issues around scalability and generalization ability of the spectral-based models by performing graph convolution operations directly in the graph domain. An increasing number of researchers have paid attention to this promising area.

Although GNNs have achieved state-of-the-art results in many tasks, most GNN-based models assume that the input is a homogeneous graph with only one node type and one edge type. Most real-world graphs consist of various types of nodes and edges associated with attributes in different feature spaces. For example, a co-authorship network contains at least two types of nodes, namely authors and papers. Author attributes may include affiliations, citations, and research fields. Paper attributes may consist of keywords, venue, year, and so on. We refer to graphs of this kind as *heterogeneous information networks* (HINs) or *heterogeneous graphs*. The heterogeneity in both graph structure and node content makes it

challenging for GNNs to encode their rich and diverse information into a low-dimensional vector space.

Most existing heterogeneous graph embedding methods are based on the idea of metapaths. A *metapath* is an ordered sequence of node types and edge types defined on the network schema, which describes a composite relation between the nodes types involved. For example, in a scholar network with authors, papers, and venues, *Author-Paper-Author* (APA) and *Author-Paper-Venue-Paper-Author* (APVPA) are metapaths describing two different relations among authors. The APA metapath associates two co-authors, while the APVPA metapath associates two authors who published papers in the same venue. Therefore, we can view a metapath as high-order proximity between two nodes. Because traditional GNNs treat all nodes equally, they are unable to model the complex structural and semantic information in heterogeneous graphs.

Although these metapath-based embedding methods outperform traditional network embedding methods on various tasks, such as node classification and link prediction, they still suffer from at least one of the following limitations. (1) The model does not leverage node content features, so it rarely performs well on heterogeneous graphs with rich node content features (e.g., metapath2vec [9], ESim [22], HIN2vec [11], and HERec [23]). (2) The model discards all intermediate nodes along the metapath by only considering two end nodes, which results in information loss (e.g., HERec [23] and HAN [31]). (3) The model relies on a single metapath to embed the heterogeneous graph. Hence, the model requires a manual metapath selection process and loses aspects of information from other metapaths, leading to suboptimal performance (e.g., metapath2vec [9]).

To address these limitations, we propose a novel *Metapath Aggregated Graph Neural Network* (MAGNN) for heterogeneous graph embedding. MAGNN addresses all the issues described above by applying node content transformation, intra-metapath aggregation, and inter-metapath aggregation to generate node embeddings. Specifically, MAGNN first applies type-specific linear transformations to project heterogeneous node attributes, with possibly unequal dimensions for different node types, to the same latent vector space. Next, MAGNN applies intra-metapath aggregation with the attention mechanism [28] for every metapath. During this intra-metapath aggregation, each target node extracts and combines information from the metapath instances connecting the node with its metapath-based neighbors. In this way, MAGNN captures the structural and semantic information of heterogeneous graphs from both neighbor nodes and the metapath context in between. Following intra-metapath aggregation, MAGNN further conducts inter-metapath aggregation using the attention mechanism to fuse latent vectors obtained from multiple metapaths into final node embeddings. By integrating multiple metapaths, our model can learn the comprehensive semantics ingrained in the heterogeneous graph.

In summary, this work makes several major contributions:

(1) We propose a novel metapath aggregated graph neural network for heterogeneous graph embedding.
(2) We design several candidate encoder functions for distilling information from metapath instances, including one based on the idea of relational rotation in complex space [24].
(3) We conduct extensive experiments on the IMDb and the DBLP datasets for node classification and node clustering, as well as on the Last.fm dataset for link prediction to evaluate the performance of our proposed model. Experiments on all of these datasets and tasks show that the node embeddings learned by MAGNN are consistently better than those generated by other state-of-the-art baselines.

Table 1: Notations used in this paper.

Notations	Definitions
$\mathbb{R}^n$	n-dimensional Euclidean space
a, $\mathbf{a}$, $\mathbf{A}$	Scalar, vector, matrix
$\mathbf{A}^\top$	Matrix/vector transpose
$\mathcal{V}$	The set of nodes in a graph
$\mathcal{E}$	The set of edges in a graph
$\mathcal{G}$	A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
v	A node $v \in \mathcal{V}$
P	A metapath
$P(v, u)$	A metapath instance connecting node v and u
$\mathcal{N}_v$	The set of neighbors of node v
$\mathcal{N}_v^P$	The set of metapath-P-based neighbors of node v
$\mathbf{x}_v$	Raw (content) feature vector of node v
$\mathbf{h}_v$	Hidden state (embedding) of node v
$\mathbf{W}$	Weight matrix
α, β	Normalized attention weight
$\sigma(\cdot)$	Activation function
$\odot$	Element-wise multiplication
$\lvert \cdot \rvert$	The cardinality of a set
$\Vert$	Vector concatenation

2 PRELIMINARY

In this section, we give formal definitions of some important terminologies related to heterogeneous graphs. Graphical illustrations are provided in Figure 1. Besides, Table 1 summarizes frequently used notations in this paper for quick reference.

Definition 2.1. Heterogeneous Graph. A heterogeneous graph is defined as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ associated with a node type mapping function $\phi : \mathcal{V} \rightarrow \mathcal{A}$ and an edge type mapping function $\psi : \mathcal{E} \rightarrow \mathcal{R}$. $\mathcal{A}$ and $\mathcal{R}$ denote the predefined sets of node types and edge types, respectively, with $|\mathcal{A}| + |\mathcal{R}| > 2$.

Definition 2.2. Metapath. A metapath P is defined as a path in the form of $A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \cdots \xrightarrow{R_l} A_{l+1}$ (abbreviated as $A_1 A_2 \cdots A_{l+1}$), which describes a composite relation $R = R_1 \circ R_2 \circ \cdots \circ R_l$ between node types A_1 and A_{l+1}, where $\circ$ denotes the composition operator on relations.

Definition 2.3. Metapath Instance. Given a metapath P of a heterogeneous graph, a metapath instance p of P is defined as a node sequence in the graph following the schema defined by P.

Definition 2.4. Metapath-based Neighbor. Given a metapath P of a heterogeneous graph, the metapath-based neighbors $\mathcal{N}_v^P$ of a node v is defined as the set of nodes that connect with node v via metapath instances of P. A neighbor connected by two different metapath instances is regarded as two different nodes in $\mathcal{N}_v^P$. Note that $\mathcal{N}_v^P$ includes v itself if P is symmetric.

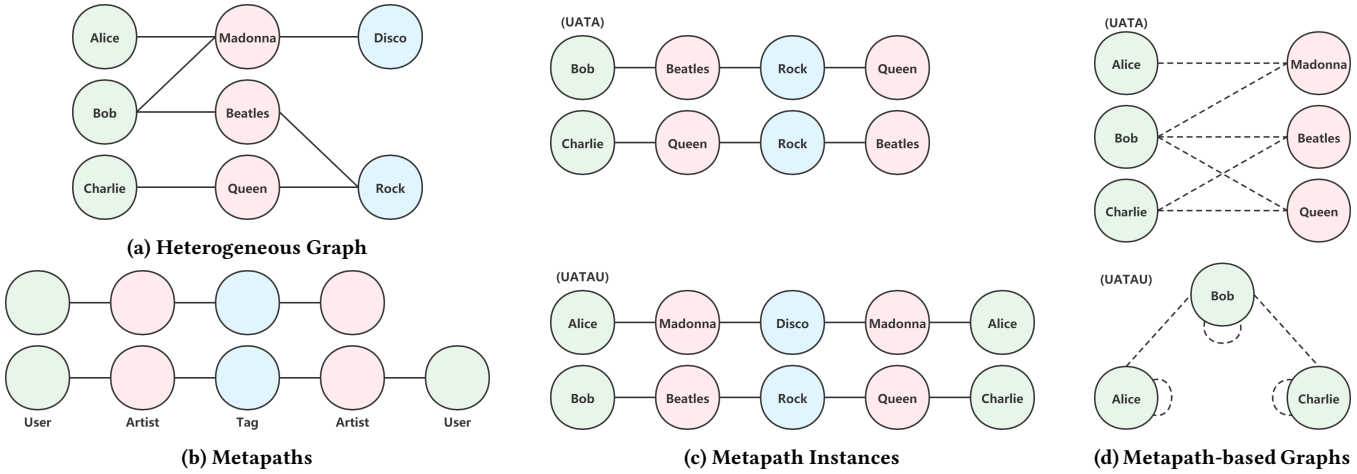


Figure 1: An illustration of the terms defined in Section 2. (a) An example heterogeneous graph with three types of nodes (i.e., users, artists, and tags). (b) The User-Artist-Tag-Artist (UATA) metapath and the User-Artist-Tag-Artist-User (UATAU) metapath. (c) Example metapath instances of the UATA and UATAU metapaths, respectively. (d) The metapath-based graphs for the UATA and UATAU metapaths, respectively.

For example, considering the metapath UATA in Figure 1, artist *Queen* is a metapath-based neighbor of user *Bob*. These two nodes are connected via the metapath instance *Bob-Beatles-Rock-Queen*. Moreover, we may refer to *Beatles* and *Rock* as the intermediate nodes along this metapath instance.

Definition 2.5. Metapath-based Graph. Given a metapath P of a heterogeneous graph $\mathcal{G}$, the metapath-based graph $\mathcal{G}^P$ is a graph constructed by all the metapath-P-based neighbor pairs in graph $\mathcal{G}$. Note that $\mathcal{G}^P$ is homogeneous if P is symmetric.

Definition 2.6. Heterogeneous Graph Embedding. Given a heterogeneous graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with node attribute matrices $\mathbf{X}_{A_i} \in \mathbb{R}^{|\mathcal{V}_{A_i}| \times d_{A_i}}$ for node types $A_i \in \mathcal{A}$, heterogeneous graph embedding is the task to learn the d-dimensional node representations $\mathbf{h}_v \in \mathbb{R}^d$ for all $v \in \mathcal{V}$ with $d \ll |\mathcal{V}|$ that are able to capture rich structural and semantic information involved in $\mathcal{G}$.

3 RELATED WORK

In this section, we review studies on graph representation learning that are related to our model. They are organized into two subsections: Section 3.1 summarizes research efforts on GNNs for general graph embedding, while Section 3.2 introduces graph embedding methods designed for heterogeneous graphs.

3.1 Graph Neural Networks

The goal of a GNN is to learn a low-dimensional vector representation $\mathbf{h}_v$ for every node v, which can be used for many downstream tasks, e.g., node classification, node clustering, and link prediction. The rationale behind this is that each node is naturally defined by its own features and its neighborhood. Following this idea and based on graph signal processing, spectral-based GNNs were first developed to perform graph convolution in the Fourier domain of a graph. ChebNet [8] utilizes Chebyshev polynomials to filter graph signals (node features) in the graph Fourier domain. Another influential model of this kind is GCN [16], which constrains and simplifies the parameters of ChebNet to alleviate the overfitting problem and improve the performance. However, spectral-based GNNs suffer from poor scalability and generalization ability, because they require the entire graph as input for every layer, and their learned filters depend on the eigenbasis of the graph Laplacian, which is closely related to the specific graph structure.

Spatial-based GNNs have been proposed to address these two limitations. GNNs of this kind define convolutions directly in the graph domain by aggregating feature information from neighbors for each node, thus imitating the convolution operations of convolutional neural networks for image data. GraphSAGE [14], the seminal spatial-based GNN framework, is founded upon the general notion of aggregator functions for efficient generation of node embeddings. The aggregator function samples, extracts, and transforms a target node's local neighborhood, and thus facilitates parallel training and generalization to unseen nodes or graphs. Many other spatial-based GNN variants have been proposed based on this idea. Inspired by the Transformer [27], GAT [28] incorporates the attention mechanism into the aggregator function to take into account the relative importance of each neighbor's information from the target node's perspective. GGNN [17] adds a gated recurrent unit (GRU) [7] to the aggregator function by treating the aggregated neighborhood information as the input to the GRU of the current time step. GaAN [34] combines GRU with the gated multi-head attention mechanism for dealing with spatiotemporal graphs. STAR-GCN [35] stacks multiple GCN encoder-decoders to boost the rating prediction performance.

All of the GNNs mentioned above are either built for homogeneous graphs, or designed for graphs with a special structure, as in user-item recommender systems. Because most existing GNNs operate on features of nodes in the same shared embedding space, they cannot be naturally adapted to heterogeneous graphs with node features lying in different spaces.

3.2 Heterogeneous Graph Embedding

Heterogeneous graph embedding aims to project nodes in a heterogeneous graph into a low-dimensional vector space. This challenging topic has been addressed by a number of studies. For example, metapath2vec [9] generates random walks guided by a single metapath, which are then fed to a skip-gram model [19] to generate node embeddings. Given user-defined metapaths, ESim [22] generates node embeddings by learning from sampled positive and negative metapath instances. HIN2vec [11] carries out multiple prediction training tasks to learn representations of nodes and metapaths of a heterogeneous graph. Given a metapath, HERec [23] converts a heterogeneous graph into a homogeneous graph based on metapath-based neighbors and applies the DeepWalk model to learn the node embeddings of the target type. Like HERec, HAN [31] converts a heterogeneous graph into multiple metapath-based homogeneous graphs in a similar way, but uses a graph attention network architecture to aggregate information from the neighbors and leverages the attention mechanism to combine various metapaths. Another model, PME [6], learns node embeddings by projecting them into the corresponding relation spaces and optimizing the proximity between the projected nodes.

However, all of the heterogeneous graph embedding methods introduced above have the limitations of either ignoring node content features, discarding all intermediate nodes along the metapath, or utilizing only a single metapath. Although they might have improved upon the performance of homogeneous graph embedding methods for some heterogeneous graph datasets, there is still room for improvement by exploiting more comprehensively the information embedded in heterogeneous graphs.

4 METHODOLOGY

In this section, we describe a new metapath aggregated graph neural network (MAGNN) for heterogeneous graph embedding. MAGNN is constructed by three major components: node content transformation, intra-metapath aggregation, and inter-metapath aggregation. Figure 2 illustrates the embedding generation of a single node. The overall forward propagation process is shown in Algorithm 1.

4.1 Node Content Transformation

For a heterogeneous graph associated with node attributes, different node types may have unequal dimensions of feature vectors. Even if they happen to be the same dimension, they may lie in different feature spaces. For example, n_1-dimensional bag-of-words vectors of texts and n_2-dimensional intensity histogram vectors of images cannot directly operate together even if $n_1 = n_2$. Feature vectors of different dimensions are troublesome when we process them in a unified framework. Therefore, we need to project different types of node features into the same latent vector space before all else.

So before feeding node vectors into MAGNN, we apply a type-specific linear transformation for each type of nodes by projecting feature vectors into the same latent factor space. For a node $v \in \mathcal{V}_A$ of type $A \in \mathcal{A}$, we have

$$\mathbf{h}'_v = \mathbf{W}_A \cdot \mathbf{x}_v^A, \tag{1}$$

where $\mathbf{x}_v \in \mathbb{R}^{d_A}$ is the original feature vector, and $\mathbf{h}'_v \in \mathbb{R}^{d'}$ is the projected latent vector of node v. $\mathbf{W}_A \in \mathbb{R}^{d' \times d_A}$ is the parametric weight matrix for type A's nodes.

The node content transformation addresses the heterogeneity of a graph that originates from the node content features. After applying this operation, all nodes' projected features share the same dimension, which facilitates the aggregation process of the next model component.

4.2 Intra-metapath Aggregation

Given a metapath P, the intra-metapath aggregation layer learns the structural and semantic information embedded in the target node, the metapath-based neighbors, and the context in between, by encoding the metapath instances of P. Let $P(v, u)$ be a metapath instance connecting the *target node* v and the *metapath-based neighbor* $u \in \mathcal{N}_v^P$, we further define the *intermediate nodes* of $P(v, u)$ as $\{m^{P(v,u)}\} = P(v, u) \setminus \{u, v\}$. Intra-metapath aggregation employs a special *metapath instance encoder* to transform all the node features along a metapath instance into a single vector,

$$\mathbf{h}_{P(v,u)} = f_\theta\left(P(v,u)\right) = f_\theta\left(\mathbf{h}'_v, \mathbf{h}'_u, \left\{\mathbf{h}'_t, \forall t \in \{m^{P(v,u)}\}\right\}\right), \tag{2}$$

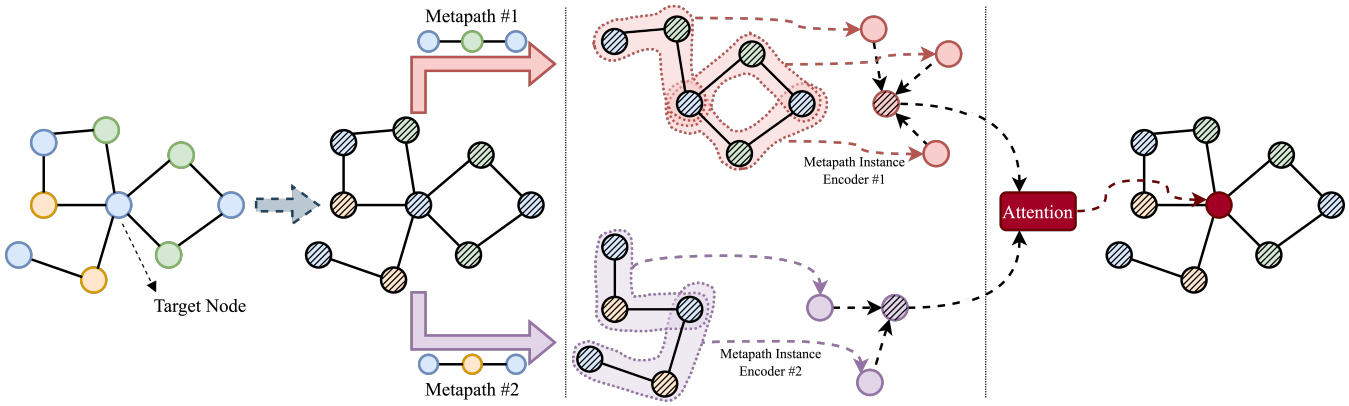
where $\mathbf{h}_{P(v,u)} \in \mathbb{R}^{d'}$ has a dimension of d'. For simplicity, here we use $P(v, u)$ to represent a single instance, although there might be multiple instances connecting the two nodes. Section 4.4 introduces several choices of a qualified metapath instance encoder.

After encoding the metapath instances into vector representations, we adopt a graph attention layer [28] to weighted sum the metapath instances of P related to target node v. The key idea is that different metapath instances would contribute to the target node's representation in different degrees. We can model this by learning a normalized importance weight α_{vu}^P for each metapath instance and then weighted summing all instances:

$$\begin{aligned} e_{vu}^P &= \text{LeakyReLU}\left(\mathbf{a}_P^\top \cdot \left[\mathbf{h}'_v \| \mathbf{h}_{P(v,u)}\right]\right), \\ \alpha_{vu}^P &= \frac{\exp\left(e_{vu}^P\right)}{\sum_{s \in \mathcal{N}_v^P} \exp\left(e_{vs}^P\right)}, \\ \mathbf{h}_v^P &= \sigma\left(\sum_{u \in \mathcal{N}_v^P} \alpha_{vu}^P \cdot \mathbf{h}_{P(v,u)}\right). \end{aligned} \tag{3}$$

Here $\mathbf{a}_P \in \mathbb{R}^{2d'}$ is the parameterized attention vector for metapath P, and $\|$ denotes the vector concatenation operator. e_{vu}^P indicates the importance of metapath instance $P(v, u)$ to node v, which is then normalized across all choices of $u \in \mathcal{N}_v^P$ using the softmax function. Once the normalized importance weight α_{vu}^P is obtained for all $u \in \mathcal{N}_v^P$, they are used to compute a weighted combination of the representations of the metapath instances about node v. Finally, the output goes through an activation function $\sigma(\cdot)$.

This attention mechanism can also be extended to multiple heads, which helps to stabilize the learning process and reduce the high variance introduced by the heterogeneity of graphs. That is, we execute K independent attention mechanisms, and then concatenate



(a) Node Content Transformation (b) Intra-metapath Aggregation (c) Inter-metapath Aggregation

Figure 2: The overall architecture of MAGNN (path instances that start and end with the target node are omitted for clarity).

their outputs, resulting in the following formulation:

$$\mathbf{h}_v^P = \Big\Vert_{k=1}^{K} \sigma\left(\sum_{u\in\mathcal{N}_v^P} \left[\alpha_{vu}^P\right]_k \cdot \mathbf{h}_{P(v,u)}\right), \tag{4}$$

where $\left[\alpha_{vu}^P\right]_k$ is the normalized importance of metapath instance $P(v,u)$ to node v at the k-th attention head.

To sum up, given the projected feature vectors $\mathbf{h}'_u \in \mathbb{R}^{d'} \forall u \in \mathcal{V}$ and the set of metapaths $\mathcal{P}_A = \{P_1, P_2, \ldots, P_M\}$ which start or end with node type $A \in \mathcal{A}$, the intra-metapath aggregation of MAGNN generates M metapath-specific vector representations of the target node $v \in \mathcal{V}_A$, denoted as $\left\{\mathbf{h}_v^{P_1}, \mathbf{h}_v^{P_2}, \ldots, \mathbf{h}_v^{P_M}\right\}$. Each $\mathbf{h}_v^{P_i} \in \mathbb{R}^{d'}$ (assuming $K = 1$) can be interpreted as a summarization of the P_i-metapath instances about node v, exhibiting one aspect of semantic information contained in node v.

4.3 Inter-metapath Aggregation

After aggregating the node and edge data within each metapath, we need to combine the semantic information revealed by all metapaths using an inter-metapath aggregation layer. Now for a node type A, we have $|\mathcal{V}_A|$ sets of latent vectors: $\left\{\mathbf{h}_v^{P_1}, \mathbf{h}_v^{P_2}, \ldots, \mathbf{h}_v^{P_M}\right\}$ for $v \in \mathcal{V}_A$, where M is the number of metapaths for type A. One straightforward inter-metapath aggregation approach is to take the element-wise mean of these node vectors. We extend this approach by exploiting the attention mechanism to assign different weights to different metapaths. This operation is reasonable because metapaths are not equally important in a heterogeneous graph.

First, we summarize each metapath $P_i \in \mathcal{P}_A$ by averaging the transformed metapath-specific node vectors for all nodes $v \in \mathcal{V}_A$,

$$\mathbf{s}_{P_i} = \frac{1}{|\mathcal{V}_A|} \sum_{v\in\mathcal{V}_A} \tanh\left(\mathbf{M}_A \cdot \mathbf{h}_v^{P_i} + \mathbf{b}_A\right), \tag{5}$$

where $\mathbf{M}_A \in \mathbb{R}^{d_m \times d'}$ and $\mathbf{b}_A \in \mathbb{R}^{d_m}$ are learnable parameters.

Then we use the attention mechanism to fuse the metapath-specific node vectors of v as follows:

$$\begin{aligned} e_{P_i} &= \mathbf{q}_A^\intercal \cdot \mathbf{s}_{P_i}, \\ \beta_{P_i} &= \frac{\exp\left(e_{P_i}\right)}{\sum_{P\in\mathcal{P}_A} \exp\left(e_P\right)}, \\ \mathbf{h}_v^{\mathcal{P}_A} &= \sum_{P\in\mathcal{P}_A} \beta_P \cdot \mathbf{h}_v^P, \end{aligned} \tag{6}$$

where $\mathbf{q}_A \in \mathbb{R}^{d_m}$ is the parameterized attention vector for node type A. β_{P_i} can be interpreted as the relative importance of metapath P_i to type A's nodes. Once β_{P_i} is computed for each $P_i \in \mathcal{P}_A$, we weighted sum all the metapath-specific node vectors of v.

At last, MAGNN employs an additional linear transformation with a nonlinear function to project the node embeddings to the vector space with the desired output dimension:

$$\mathbf{h}_v = \sigma\left(\mathbf{W}_o \cdot \mathbf{h}_v^{\mathcal{P}_A}\right), \tag{7}$$

where $\sigma(\cdot)$ is an activation function, and $\mathbf{W}_o \in \mathbb{R}^{d_o \times d'}$ is a weight matrix. This projection is task-specific. It can be interpreted as a linear classifier for node classification or regarded as a projection to the space with node similarity measures for link prediction.

4.4 Metapath Instance Encoders

To encode each metapath instance in Section 4.2, we examine three candidate encoder functions:

- **Mean encoder**. This function takes the element-wise mean of the node vectors along the metapath instance $P(v,u)$:

$$\mathbf{h}_{P(v,u)} = \text{MEAN}\left(\left\{\mathbf{h}'_t, \forall t \in P(v,u)\right\}\right). \tag{8}$$

- **Linear encoder**. This function is an extension to the mean encoder by appending it with a linear transformation:

$$\mathbf{h}_{P(v,u)} = \mathbf{W}_P \cdot \text{MEAN}\left(\left\{\mathbf{h}'_t, \forall t \in P(v,u)\right\}\right). \tag{9}$$

- **Relational rotation encoder**. We also examine a metapath instance encoder based on relational rotation in complex space, an operation proposed by RotatE [24] for knowledge graph embedding. The mean and linear encoders introduced above treat the metapath instance essentially as a set, and thus ignore the information embedded in the sequential structure of the metapath. Relational rotation provides a way to model this kind of knowledge. Given $P(v,u) = (t_0, t_1, \ldots, t_n)$ with $t_0 = u$ and $t_n = v$, let R_i be the relation between node t_{i-1} and node t_i, let $\mathbf{r}_i$ be the relation vector of R_i, the relational rotation encoder is formulated as:

$$\begin{aligned} \mathbf{o}_0 &= \mathbf{h}'_{t_0} = \mathbf{h}'_u, \\ \mathbf{o}_i &= \mathbf{h}'_{t_i} + \mathbf{o}_{i-1} \odot \mathbf{r}_i, \\ \mathbf{h}_{P(v,u)} &= \frac{\mathbf{o}_n}{n+1}, \end{aligned} \tag{10}$$

  where $\mathbf{h}'_{t_i}$ and $\mathbf{r}_i$ are both complex vectors, $\odot$ is the element-wise product. We can easily interpret a real vector of dimension d' as a complex vector of dimension $d'/2$ by treating the first half of the vector as the real part, and the second half as the imaginary part.

4.5 Training

After applying components introduced in the previous sections, we obtain the final node representations, which can then be used in different downstream tasks. Depending on the characteristics of different tasks and the availability of node labels, we can train MAGNN in two major learning paradigms, i.e., semi-supervised learning and unsupervised learning.

For semi-supervised learning, with the guide of a small fraction of labeled nodes, we can optimize the model weights by minimizing the cross entropy via backpropagation and gradient descent, and thereby learn meaningful node embeddings for heterogeneous graphs. The cross entropy loss for this semi-supervised learning is formulated as:

$$\mathcal{L} = -\sum_{v \in \mathcal{V}_L} \sum_{c=1}^{C} \mathbf{y}_v[c] \cdot \log \mathbf{h}_v[c], \tag{11}$$

where $\mathcal{V}_L$ is the set of nodes that have labels, C is the number of classes, $\mathbf{y}_v$ is the one-hot label vector of node v, and $\mathbf{h}_v$ is the predicted probability vector of node v.

For unsupervised learning, without any node labels, we can optimize the model weights by minimizing the following loss function through negative sampling [20]:

$$\mathcal{L} = -\sum_{(u,v) \in \Omega} \log \sigma\left(\mathbf{h}_u^\intercal \cdot \mathbf{h}_v\right) - \sum_{(u',v') \in \Omega^-} \log \sigma\left(-\mathbf{h}_{u'}^\intercal \cdot \mathbf{h}_{v'}\right), \tag{12}$$

where $\sigma(\cdot)$ is the sigmoid function, Ω is the set of observed (positive) node pairs, Ω^- is the set of negative node pairs sampled from all unobserved node pairs (the complement of Ω).

5 EXPERIMENTS

In this section, we present experiments to demonstrate the efficacy of MAGNN for heterogeneous graph embedding. The experiments aim to address the following research questions:

- RQ1. How does MAGNN perform in classifying nodes?
- RQ2. How does MAGNN perform in clustering nodes?
- RQ3. How does MAGNN perform in predicting plausible links between node pairs?
- RQ4. What is the impact of the three major components of MAGNN described in the previous section?
- RQ5. How do we understand the representation capability of different graph embedding methods?

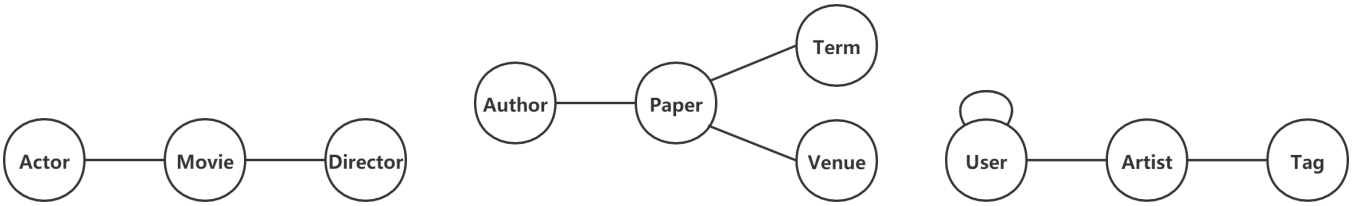
Algorithm 1: MAGNN forward propagation.

Input: The heterogeneous graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$,
node types $\mathcal{A} = \{A_1, A_2, \ldots, A_{|\mathcal{A}|}\}$,
metapaths $\mathcal{P} = \{P_1, P_2, \ldots P_{|\mathcal{P}|}\}$,
node features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$,
the number of attention heads K,
the number of layers L
Output: The node embeddings $\{\mathbf{z}_v, \forall v \in \mathcal{V}\}$

for *node type* $A \in \mathcal{A}$ **do**
  Node content transformation $\mathbf{h}_v^0 \leftarrow \mathbf{W}_A \cdot \mathbf{x}_v, \forall v \in \mathcal{V}_A$;
end
for $l = 1 \ldots L$ **do**
  for *node type* $A \in \mathcal{A}$ **do**
    for *metapath* $P \in \mathcal{P}_A$ **do**
      for $v \in \mathcal{V}_A$ **do**
        Calculate $\mathbf{h}^l_{P(v,u)}$ for all $u \in \mathcal{N}_v^P$ using the metapath instance encoder function;
        Combine extracted metapath instances $\left[\mathbf{h}_v^P\right]^l \leftarrow \Big\Vert_{k=1}^{K} \sigma\left(\sum_{u \in \mathcal{N}_v^P} \left[\alpha_{vu}^P\right]_k \cdot \mathbf{h}^l_{P(v,u)}\right)$;
      end
    end
    Calculate the weight β_P for each metapath $P \in \mathcal{P}_A$;
    Fuse the embeddings from different metapaths $\left[\mathbf{h}_v^{\mathcal{P}_A}\right]^l \leftarrow \sum_{P \in \mathcal{P}_A} \beta_P \cdot \left[\mathbf{h}_v^P\right]^l, \forall v \in \mathcal{V}_A$;
  end
  Layer output projection $\mathbf{h}_v^l = \sigma\left(\mathbf{W}_o^l \cdot \left[\mathbf{h}_v^{\mathcal{P}_A}\right]^l\right), \forall v \in \mathcal{V}_A, \forall A \in \mathcal{A}$;
end
$\mathbf{z}_v \leftarrow \mathbf{h}_v^L, \forall v \in \mathcal{V}$;

5.1 Datasets

We adopt three widely used heterogeneous graph datasets from different domains to evaluate the performance of MAGNN as compared to state-of-the-art baselines. Specifically, the IMDb and DBLP datasets are used in the experiments for node classification and node clustering. The Last.fm dataset is used in the experiments for link prediction. Simple statistics of the three datasets are summarized in Table 2, and network schemas are illustrated in Figure 3. We assign one-hot id vectors to nodes with no attributes as their dummy input features.



(a) IMDb (b) DBLP (c) Last.fm

Figure 3: Network schemas of the three heterogeneous graph datasets used in this paper.

- **IMDb**[1] is an online database about movies and television programs, including information such as cast, production crew, and plot summaries. We use a subset of IMDb scraped from online, containing 4278 movies, 2081 directors, and 5257 actors after data preprocessing. Movies are labeled as one of three classes (*Action*, *Comedy*, and *Drama*) based on their genre information. Each movie is also described by a bag-of-words representation of its plot keywords. For semi-supervised learning models, the movie nodes are divided into training, validation, and testing sets of 400 (9.35%), 400 (9.35%), and 3478 (81.30%) nodes, respectively.
- **DBLP**[2] is a computer science bibliography website. We adopt a subset of DBLP extracted by [12, 15], containing 4057 authors, 14328 papers, 7723 terms, and 20 publication venues after data preprocessing. The authors are divided into four research areas (*Database*, *Data Mining*, *Artificial Intelligence*, and *Information Retrieval*). Each author is described by a bag-of-words representation of their paper keywords. For semi-supervised learning models, the author nodes are divided into training, validation, and testing sets of 400 (9.86%), 400 (9.86%), and 3257 (80.28%) nodes, respectively.
- **Last.fm**[3] is a music website keeping track of users' listening information from various sources. We adopt a dataset released by HetRec 2011 [4], consisting of 1892 users, 17632 artists, and 1088 artist tags after data preprocessing. This dataset is used for the link prediction task, and no label or feature is included in this dataset. For semi-supervised learning models, the user-artist pairs are divided into training, validation, and testing sets of 64984 (70%), 9283 (10%), and 18567 (20%) pairs, respectively.

Table 2: Statistics of datasets.

Dataset	Node	Edge	Metapath
IMDb	# movie (M): 4,278 # director (D): 2,081 # actor (A): 5,257	# M-D: 4,278 # M-A: 12,828	MDM MAM DMD DMAMD AMA AMDMA
DBLP	# author (A): 4,057 # paper (P): 14,328 # term (T): 7,723 # venue (V): 20	# A-P: 19,645 # P-T: 85,810 # P-V: 14,328	APA APTPA APVPA
Last.fm	# user (U): 1,892 # artist (A): 17,632 # tag (T): 1,088	# U-U: 12,717 # U-A: 92,834 # A-T: 23,253	UU UAU UATAU AUA AUUA ATA

5.2 Baselines

We compare MAGNN against different kinds of graph embedding models, including traditional (as opposed to GNNs) homogeneous graph embedding models, traditional heterogeneous graph embedding models, GNNs for homogeneous graphs, and GNNs for heterogeneous graphs. We denote them as *traditional homogeneous models*, *traditional heterogeneous models*, *homogeneous GNNs*, and *heterogeneous GNNs*, respectively. The list of baseline models is shown as follows.

- **LINE** [25] is a *traditional homogeneous model* exploiting the first-order and second-order proximity between nodes. We apply it to the heterogeneous graphs by ignoring the heterogeneity of graph structure and dropping all node content features. The LINE variant using second-order proximity is applied in our experiments.
- **node2vec** [13] is a *traditional homogeneous model* serving as a generalized version of DeepWalk [21]. We apply it to the heterogeneous graphs in the same way as LINE.
- **ESim** [22] is a *traditional heterogeneous model* that learns node embeddings from sampled metapath instances. ESim requires a predefined weight for each metapath. Here we assign equal weights to all metapaths because searching for the optimal weights of metapaths is difficult, and does not provide a significant performance gain over equal weights according to the authorsâĂŹ experiments.
- **metapath2vec** [9] is a *traditional heterogeneous model* that generates node embeddings by feeding metapath-guided random walks to a skip-gram model. This model relies on a single user-specified metapath, so we test on all metapaths separately and report the one with the best results. We use the metapath2vec++ model variant in our experiments.

[1] https://www.imdb.com/
[2] https://dblp.uni-trier.de/
[3] https://www.last.fm/

Table 3: Experiment results (%) on the IMDb and DBLP datasets for the node classification task.

Dataset	Metrics	Train %	Unsupervised					Semi-supervised			
			LINE	node2vec	ESim	metapath2vec	HERec	GCN	GAT	HAN	MRGNN
IMDb	Macro-F1	20%	44.04	49.00	48.37	46.05	45.61	52.73	53.64	56.19	**59.35**
		40%	45.45	50.63	50.09	47.57	46.80	53.67	55.50	56.15	**60.27**
		60%	47.09	51.65	51.45	48.17	46.84	54.24	56.46	57.29	**60.66**
		80%	47.49	51.49	51.37	49.99	47.73	54.77	57.43	58.51	**61.44**
	Micro-F1	20%	45.21	49.94	49.32	47.22	46.23	52.80	53.64	56.32	**59.60**
		40%	46.92	51.77	51.21	48.17	47.89	53.76	55.56	57.32	**60.50**
		60%	48.35	52.79	52.53	49.87	48.19	54.23	56.47	58.42	**60.88**
		80%	48.98	52.72	52.54	50.50	49.11	54.63	57.40	59.24	**61.53**
DBLP	Macro-F1	20%	87.16	86.70	90.68	88.47	90.82	88.00	91.05	91.69	**93.13**
		40%	88.85	88.07	91.61	89.91	91.44	89.00	91.24	91.96	**93.23**
		60%	88.93	88.69	91.84	90.50	92.08	89.43	91.42	92.14	**93.57**
		80%	89.51	88.93	92.27	90.86	92.25	89.98	91.73	92.50	**94.10**
	Micro-F1	20%	87.68	87.21	91.21	89.02	91.49	88.51	91.61	92.33	**93.61**
		40%	89.25	88.51	92.05	90.36	92.05	89.22	91.77	92.57	**93.68**
		60%	89.34	89.09	92.28	90.94	92.66	89.57	91.97	92.72	**93.99**
		80%	89.96	89.37	92.68	91.31	92.78	90.33	92.24	93.23	**94.47**

- **HERec** [23] is a *traditional heterogeneous model* that learns node embeddings by applying DeepWalk to the metapath-based homogeneous graphs converted from the original heterogeneous graph. This model comes with an embedding fusion algorithm designed for rating prediction, which can be adapted to link prediction. For node classification/clustering, we select and report the metapath with the best performance.
- **GCN** [16] is a *homogeneous GNN*. This model performs convolutional operations in the graph Fourier domain. Here we test GCN on metapath-based homogeneous graphs and report the results from the best metapath.
- **GAT** [28] is a *homogeneous GNN*. This model performs convolutional operations in the graph spatial domain with the attention mechanism incorporated. Similarly, here we test GAT on metapath-based homogeneous graphs and report the results from the best metapath.
- **GATNE** [5] is a *heterogeneous GNN*. It generates a node's representation from the base embedding and the edge embeddings, with a focus on the link prediction task. Here we report the results from the best-performing GATNE variant.
- **HAN** [31] is a *heterogeneous GNN*. It learns metapath-specific node embeddings from different metapath-based homogeneous graphs, and leverages the attention mechanism to combine them into one vector representation for each node.

For traditional models, including LINE, node2vec, ESim, metapath2vec, and HERec, we set the window size to 5, walk length to 100, walks per node to 40, and number of negative samples to 5, if applicable. For GNNs, including GCN, GAT, HAN, and our proposed MAGNN, we set the dropout rate to 0.5; we use the same splits of training, validation, and testing sets; we employ the Adam optimizer with the learning rate set to 0.005 and the weight decay (L2 penalty) set to 0.001; we train the GNNs for 100 epochs and apply early stopping with a patience of 30. For node classification and node clustering, the GNNs are trained in a semi-supervised fashion with a small fraction of nodes labeled as guidance. For GAT, HAN, and MAGNN, we set the number of attention heads to 8. For HAN and MAGNN, we set the dimension of the attention vector in inter-metapath aggregation to 128. For a fair comparison, we set the embedding dimension of all the models mentioned above to 64.

5.3 Node Classification (RQ1)

We conduct experiments on the IMDb and DBLP datasets to compare the performance of different models on the node classification task. We feed the embeddings of labeled nodes (movies in IMDb and authors in DBLP) generated by each learning model to a linear support vector machine (SVM) classifier with varying training proportions. Note that for a fair comparison, only the nodes in the testing set are fed to the linear SVM, because semi-supervised models have already "seen" the nodes in the training and validation sets, as shown in Equation 11. Hence, the training and testing proportions of the linear SVM here only concern the testing set (i.e., 3478 nodes for IMDb and 3257 nodes for DBLP). Again, the train/test splits for the linear SVM are also the same across embedding models. Similar strategies are also applied to the experiments of node clustering and link prediction. We report the average *Macro-F1* and *Micro-F1* of 10 runs of each embedding model in Table 3.

As shown in the table, MAGNN performs consistently better than other baselines across different training proportions and datasets. On IMDb, it is interesting to see that node2vec performs better than traditional heterogeneous models. That said, GNNs, especially heterogeneous GNNs, obtain even better results, demonstrating that the GNN architecture, which judiciously utilizes the heterogeneous node features, helps improve the embedding performance. The performance gain obtained by MAGNN over the best baseline (HAN) is around 4-7%, which indicates that metapath instances contain richer information than metapath-based neighbors. On DBLP, the node classification task is trivial, as evident from the high scores of all models. Even so, MAGNN still outperforms the strongest baseline by 1-2%.

Table 4: Experiment results (%) on the IMDb and DBLP datasets for the node clustering task.

Dataset	Metrics	Unsupervised					Semi-supervised			
		LINE	node2vec	ESim	metapath2vec	HERec	GCN	GAT	HAN	MRGNN
IMDb	NMI	1.13	5.22	1.07	0.89	0.39	7.46	7.84	10.79	**15.58**
	ARI	1.20	6.02	1.01	0.22	0.11	7.69	8.87	11.11	**16.74**
DBLP	NMI	71.02	77.01	68.33	74.18	69.03	73.45	70.73	77.49	**80.81**
	ARI	76.52	81.37	72.22	78.11	72.45	77.50	76.04	82.95	**85.54**

Table 5: Experiment results (%) on the Last.fm dataset for the link prediction task.

Dataset	Metrics	LINE	node2vec	ESim	metapath2vec	HERec	GCN	GAT	GATNE	HAN	MAGNN
Last.fm	AUC	85.76	67.14	82.00	92.20	91.52	90.97	92.36	89.21	93.40	**98.91**
	AP	88.07	64.11	82.19	90.11	89.47	91.65	91.55	88.86	92.44	**98.93**

5.4 Node Clustering (RQ2)

We conduct experiments on the IMDb and DBLP datasets to compare the performance of different models on the node clustering task. We feed the embeddings of labeled nodes (movies in IMDb and authors in DBLP) generated by each learning model to the K-Means algorithm. The number of clusters in K-Means is set to the number of classes for each dataset, i.e., 3 for IMDb and 4 for DBLP. We employ the *normalized mutual information* (NMI) and *adjusted Rand index* (ARI) as the evaluation metrics. Since the clustering result of the K-Means algorithm is highly dependent on the initialization of the centroids, we repeat K-Means 10 times for each run of the embedding model, and each embedding model is tested for 10 runs. We report the averaged results in Table 4.

From Table 4, we can see that MAGNN is consistently superior to all other baselines in node clustering. Note that all models have much poorer performance on IMDb than on DBLP. This is presumably because of the dirty labels of movies in IMDb: every movie node in the original IMDb dataset has multiple genres, and we only choose the very first one as its class label. We can see that the traditional heterogeneous models do not have many advantages over the traditional homogeneous models in node clustering. Node2vec is expected to perform strongly in the node clustering task because, being a random-walk-based approach, it forces nodes that are close in the graph also to be close in the embedding space [33], and thereby encodes node positional information. This property implicitly facilitates the K-Means algorithm as it clusters nodes based on the Euclidean distances between embeddings. Despite this, the heterogeneity-aware GNNs (i.e., HAN and MAGNN) still rank the first in node clustering on both datasets.

5.5 Link Prediction (RQ3)

We also conduct experiments on the Last.fm dataset to evaluate the performance of MAGNN and other baselines in the link prediction task. For the GNNs, we treat the connected user-artist pair as positive node pairs, and consider all unconnected user-artist links as negative node pairs. We add the same number of randomly sampled negative node pairs to the validation and testing sets. During the GNNs' training, negative node pairs are also uniformly sampled on the fly. The GNNs are then optimized by minimizing the objective function described in Equation 12.

Given the user embedding $\mathbf{h}_u$ and the artist embedding $\mathbf{h}_a$ generated by the trained model, we calculate the probability that u and v link together as follows:

$$p_{ua} = \sigma\left(\mathbf{h}_u^\top \cdot \mathbf{h}_a\right), \tag{13}$$

where $\sigma(\cdot)$ is the sigmoid function. The embedding models for link prediction are evaluated by the *area under the ROC curve* (AUC) and *average precision* (AP) scores. We report the averaged results of 10 runs of each embedding model in Table 5.

From Table 5, MAGNN outperforms other baseline models by a large margin. The strongest traditional model here is metapath2vec, which learns from node sequences generated from random walks guided by a single metapath. MAGNN achieves better scores than metapath2vec, showing that considering a single metapath is suboptimal. Among GNN baselines, HAN obtains the best results because it is heterogeneity-aware and combines multiple metapaths. Our MAGNN achieves a relative improvement of around 6% over HAN. This result supports our claim that the metapath contexts of nodes are critical to the node embeddings.

5.6 Ablation Study (RQ4)

To validate the effectiveness of each component of our model, we further conduct experiments on different MAGNN variants. Here we report the results obtained from the three datasets on all three tasks in Table 6. Note that every presented score of the node classification task (i.e., Macro-F1 and Micro-F1) is an average of the scores in different training proportions (explained in Section 5.3). Here $\text{MAGNN}_{\text{rot}}$ is our proposed model using the relational rotation encoder, i.e., the one used to compete with other baselines in Table 3, 4, and 5. Let $\text{MAGNN}_{\text{rot}}$ be the reference model, $\text{MAGNN}_{\text{feat}}$ is the equivalent model without utilizing node content features; MAGNN_{nb} considers only the metapath-based neighbors; MAGNN_{sm} considers the single best metapath; $\text{MAGNN}_{\text{avg}}$ switches to using the mean metapath instance encoder; $\text{MAGNN}_{\text{linear}}$ switches to using the linear metapath instance encoder. Except for the above-mentioned differences, all other settings are the same for these MAGNN variants. Note that $\text{MAGNN}_{\text{feat}}$ on Last.fm is equivalent to $\text{MAGNN}_{\text{rot}}$ because this dataset does not contain node attributes.

As can be seen, by utilizing the node content features, $\text{MAGNN}_{\text{rot}}$ obtains a significant performance improvement over $\text{MAGNN}_{\text{feat}}$,

Table 6: Quantitative results (%) for ablation study.

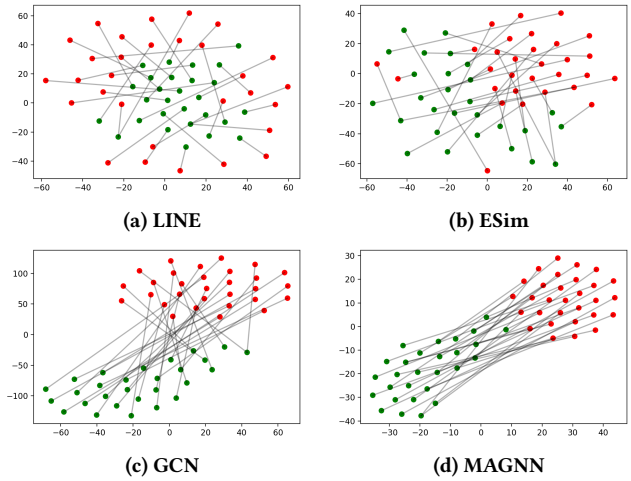
Variant	IMDb				DBLP				Last.fm	
	Macro-F1	Micro-F1	NMI	ARI	Macro-F1	Micro-F1	NMI	ARI	AUC	AP
$\text{MAGNN}_{\text{feat}}$	48.87	50.36	5.82	5.30	92.80	93.32	77.17	82.15	N/A	N/A
MAGNN_{nb}	58.45	58.84	12.87	11.98	92.61	93.15	77.64	82.60	93.68	92.95
MAGNN_{sm}	56.77	56.64	11.90	11.84	93.19	93.69	79.48	84.39	92.54	91.52
$\text{MAGNN}_{\text{avg}}$	59.66	59.78	13.64	15.27	93.13	93.44	79.31	84.30	98.63	98.57
$\text{MAGNN}_{\text{linear}}$	57.80	57.96	9.80	8.49	93.21	93.52	78.95	83.89	98.56	98.48
$\text{MAGNN}_{\text{rot}}$	**60.43**	**60.63**	**15.58**	**16.74**	**93.51**	**93.94**	**80.81**	**85.54**	**98.91**	**98.93**

which shows the necessity of applying node content transformation to incorporate node features. Comparing MAGNN_{nb} with $\text{MAGNN}_{\text{avg}}$, $\text{MAGNN}_{\text{linear}}$, and $\text{MAGNN}_{\text{rot}}$, we see that aggregating metapath instances rather than metapath-based neighbors brings about a boost in performance, which validates the efficacy of intra-metapath aggregation. Next, the difference between the results of MAGNN_{sm} and $\text{MAGNN}_{\text{rot}}$ reveals that the model performance is improved considerably by combining multiple metapaths in inter-metapath aggregation. Finally, the results of $\text{MAGNN}_{\text{avg}}$, $\text{MAGNN}_{\text{linear}}$, and $\text{MAGNN}_{\text{rot}}$ suggest that the relational rotation encoder does help to improve MAGNN by a small margin. It is interesting to see that $\text{MAGNN}_{\text{linear}}$ performs worse than $\text{MAGNN}_{\text{avg}}$. Nonetheless, all three MAGNN variants using different encoders still consistently outperform the best baseline, HAN.

5.7 Visualization (RQ5)

In addition to the quantitative evaluations of embedding models, we also visualize node embeddings to conduct a qualitative assessment of the embedding results. We randomly select 30 user-artist pairs from the positive testing set of the Last.fm dataset, and then project the embeddings of these nodes into a 2-dimensional space using t-SNE. Here we illustrate the visualization results of LINE, ESim, GCN, and MAGNN in Figure 4, where red points and green points indicate users and artists, respectively.

Based on this visualization, one can quickly tell the differences among graph embedding models in terms of their learning ability towards heterogeneous graphs. As a traditional homogeneous graph embedding model, LINE cannot effectively divide user nodes and artist nodes into two different groups. In contrast, ESim, a traditional heterogeneous model, can roughly partition the two types of nodes. Thanks to the powerful GNN architecture and by choosing appropriate metapaths, a homogeneous GNN such as GCN can isolate different types of nodes and encode the correlation information of the user-artist pairs into the node embeddings. From Figure 4, we can see that our proposed MAGNN obtains the best embedding results, with two well-separated user and artist groups, and an aligned correlation of user-artist pairs.



(a) LINE **(b) ESim** **(c) GCN** **(d) MAGNN**

Figure 4: Embedding visualization of node pairs in Last.fm.

6 CONCLUSION

In this paper, we propose a novel metapath aggregated graph neural network (MAGNN) to address the three characteristic limitations of existing heterogeneous graph embedding methods, namely (1) dropping node content features, (2) discarding intermediate nodes along metapaths, and (3) considering only a single metapath. To be specific, MAGNN applies three building block components: (1) node content transformation, (2) intra-metapath aggregation, and (3) inter-metapath aggregation to deal with each of the limitations, respectively. Additionally, we define the notion of metapath instance encoders, which are used to extract the structural and semantic information ingrained in metapath instances. We propose several candidate encoder functions, including one inspired by the RotatE knowledge graph embedding model [24]. In experiments, MAGNN achieves state-of-the-art results on three real-world datasets in the node classification, node clustering, and link prediction tasks. Ablation studies also demonstrate the efficacy of the three major components of MAGNN in boosting embedding performance. We plan to adapt this heterogeneous graph embedding framework to the rating prediction (recommendation) task with the user-item data assisted by the heterogeneous knowledge graph [30].

ACKNOWLEDGMENTS

The work described in this paper was partially supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (CUHK 2300174 (Collaborative Research Fund, No. C5026-18GF) and CUHK 3133238 (Research Sustainability of Major RGC Funding Schemes)).

REFERENCES

[1] James Atwood and Don Towsley. 2016. Diffusion-Convolutional Neural Networks. In *NIPS*. 1993–2001.

[2] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and koray kavukcuoglu. 2016. Interaction Networks for Learning about Objects, Relations and Physics. In *NIPS*. 4502–4510.

[3] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *NIPS*. 2787–2795.

[4] Iván Cantador, Peter Brusilovsky, and Tsvi Kuflik. 2011. 2nd Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec 2011). In *RecSys*.

[5] Yukuo Cen, Xu Zou, Jianwei Zhang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Representation Learning for Attributed Multiplex Heterogeneous Network. In *SIGKDD*. 1358–1368.

[6] Hongxu Chen, Hongzhi Yin, Weiqing Wang, Hao Wang, Quoc Viet Hung Nguyen, and Xue Li. 2018. PME: Projected Metric Embedding on Heterogeneous Networks for Link Prediction. In *SIGKDD*. 1177–1186.

[7] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *CoRR* abs/1406.1078 (2014). arXiv:1406.1078

[8] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *NIPS*. 3844–3852.

[9] Yuxiao Dong, Nitesh V. Chawla, and Ananthram Swami. 2017. Metapath2Vec: Scalable Representation Learning for Heterogeneous Networks. In *SIGKDD*. 135–144.

[10] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. 2017. Protein Interface Prediction using Graph Convolutional Networks. In *NIPS*. 6530–6539.

[11] Tao-yang Fu, Wang-Chien Lee, and Zhen Lei. 2017. HIN2Vec: Explore Meta-paths in Heterogeneous Information Networks for Representation Learning. In *CIKM*. 1797–1806.

[12] Jing Gao, Feng Liang, Wei Fan, Yizhou Sun, and Jiawei Han. 2009. Graph-based Consensus Maximization Among Multiple Supervised and Unsupervised Models. In *NIPS*. 585–593.

[13] Aditya Grover and Jure Leskovec. 2016. Node2Vec: Scalable Feature Learning for Networks. In *SIGKDD*. 855–864.

[14] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NIPS*. 1024–1034.

[15] Ming Ji, Yizhou Sun, Marina Danilevsky, Jiawei Han, and Jing Gao. 2010. Graph Regularized Transductive Classification on Heterogeneous Information Networks. In *ECML PKDD*. 570–586.

[16] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[17] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. 2016. Gated Graph Sequence Neural Networks. In *ICLR*.

[18] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *ICLR*.

[19] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *ICLR*.

[20] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*. 3111–3119.

[21] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online Learning of Social Representations. In *SIGKDD*. 701–710.

[22] Jingbo Shang, Meng Qu, Jialu Liu, Lance M. Kaplan, Jiawei Han, and Jian Peng. 2016. Meta-Path Guided Embedding for Similarity Search in Large-Scale Heterogeneous Information Networks. *CoRR* abs/1610.09769 (2016). arXiv:1610.09769

[23] Chuan Shi, Binbin Hu, Wayne Xin Zhao, and Philip S. Yu. 2019. Heterogeneous Information Network Embedding for Recommendation. *IEEE Transactions on Knowledge and Data Engineering* 31, 2 (2019), 357–370.

[24] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In *ICLR*.

[25] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding. In *WWW*. 1067–1077.

[26] Rianne van den Berg, Thomas N. Kipf, and Max Welling. 2017. Graph Convolutional Matrix Completion. *CoRR* abs/1706.02263 (2017). arXiv:1706.02263

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NIPS*. 5998–6008.

[28] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.

[29] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural Deep Network Embedding. In *SIGKDD*. 1225–1234.

[30] Hongwei Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. 2019. Knowledge Graph Convolutional Networks for Recommender Systems. In *WWW*. 3307–3313.

[31] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous Graph Attention Network. In *WWW*. 2022–2032.

[32] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y. Chang. 2015. Network Representation Learning with Rich Text Information. In *IJCAI*. 2111–2117.

[33] Jiaxuan You, Rex Ying, and Jure Leskovec. 2019. Position-aware Graph Neural Networks. In *ICML*. 7134–7143.

[34] Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit-Yan Yeung. 2018. GaAN: Gated Attention Networks for Learning on Large and Spatiotemporal Graphs. In *UAI*. 339–349.

[35] Jiani Zhang, Xingjian Shi, Shenglin Zhao, and Irwin King. 2019. STAR-GCN: Stacked and Reconstructed Graph Convolutional Networks for Recommender Systems. In *IJCAI*. 4264–4270.