Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2023.024781

REVIEW

Heterogeneous Network Embedding: A Survey

Sufen Zhao[1,2], Rong Peng[1,*], Po Hu[2] and Liansheng Tan[2]

[1]School of Computer Science, Wuhan University, Wuhan, 430072, China

[2]School of Computer Science, Central China Normal University, Wuhan, 430079, China

*Corresponding Author: Rong Peng. Email: rongpeng@whu.edu.cn

Received: 08 June 2022 Accepted: 06 December 2022

ABSTRACT

Real-world complex networks are inherently heterogeneous; they have different types of nodes, attributes, and relationships. In recent years, various methods have been proposed to automatically learn how to encode the structural and semantic information contained in heterogeneous information networks (HINs) into low-dimensional embeddings; this task is called heterogeneous network embedding (HNE). Efficient HNE techniques can benefit various HIN-based machine learning tasks such as node classification, recommender systems, and information retrieval. Here, we provide a comprehensive survey of key advancements in the area of HNE. First, we define an encoder-decoder-based HNE model taxonomy. Then, we systematically overview, compare, and summarize various state-of-the-art HNE models and analyze the advantages and disadvantages of various model categories to identify more potentially competitive HNE frameworks. We also summarize the application fields, benchmark datasets, open source tools, and performance evaluation in the HNE area. Finally, we discuss open issues and suggest promising future directions. We anticipate that this survey will provide deep insights into research in the field of HNE.

KEYWORDS

Heterogeneous information networks; representation learning; heterogeneous network embedding; graph neural networks; machine learning

1 Introduction

Real-world complex networks, such as social, biological protein, and computer networks, are inherently heterogeneous. These networks, called heterogeneous information networks (HINs), contain various types of nodes, attributes, and relationships [1]. In recent years, with the rapid development of artificial intelligence, HIN analysis has attracted significant research attention. Efficient analysis techniques can benefit machine learning (ML) tasks based on HINs, such as node classification, community detection, and recommender systems.

In many HIN-based ML models, the core task is to find a way to convert the structural and semantic information of an HIN into low-dimensional vectors and then input these vectors into various downstream machine learning tasks. Feature engineering, which is the traditional approach used for feature extraction, is inefficient and highly dependent on the experience of engineers. With

the successful application of the Word2vec model [2,3] in the field of natural language processing, many recent studies have used ML methods to automatically learn the low-dimensional features of nodes, edges, and subgraphs from networks such that the obtained feature vectors can capture as much structural, semantic, and attribute information in the HINs as possible. This process is called heterogeneous network embedding (HNE). In contrast to the traditional approach, HNE has the advantages of high efficiency and compression. More importantly, most HNE models can learn features from data in a completely unsupervised manner. They do not require labeled data; instead, they generate high-quality feature representations by creating labels from the data themselves using context-based, time series-based, and contrast-based methods, among others. To make the learned feature representation more effective for specific applications, the features generated by unsupervised learning can be input as pretraining parameters to subsequent specific ML tasks, and then fine-tuned using labeled data. Alternatively, the unsupervised and supervised parts can be fused into a model for end-to-end training. When a large amount of manually labeled information is difficult to obtain, HNE can reduce the model's dependence on labeled information. Therefore, research on HNE is of great significance for artificial intelligence-related applications.

The primary goal of an HNE model is to enable the generated network embeddings to reconstruct various types of information contained in the HINs such that they can be easily used in downstream ML tasks. However, because of the complexity of HINs, existing HNE research often faces the following challenges: 1) **Heterogeneity**. In an HIN, different types of nodes usually have different types of attributes (multi-modality), and different types of nodes usually establish different types of semantic relationships (multiplex). Numerous existing network embedding models operate on homogeneous networks and are difficult to be extended to HINs. 2) **Large-scale**. In the real world, HINs are usually large and hyperscale, containing thousands of nodes and complex relationships. Many existing network embedding models are only able to run on small networks and cannot scale to large-scale networks. 3) **Dynamism**. In the real world, complex heterogeneous networks tend to constantly change. New nodes join and old nodes exit, and new links may be generated anytime and anywhere. Most traditional network embedding models are designed for static snapshot networks and cannot capture the dynamic characteristics of real-time networks. 4) **Incomplete data and noise**. Real-world network data are often incomplete and noisy. Many existing models do not consider robustness issues, resulting in brittle models and compromised performance. 5) **Multi-objectiveness**. A good HNE model usually needs to consider multiple modeling goals; such as capturing the local and global structural features of the network, capturing diverse semantic information, attribute information and label information. All the factors outlined above pose serious challenges to research in the field of HNE.

To tackle the issues cited above, numerous HNE models have been proposed over the past several years, such as the meta-path based random walk model metapath2vec [4], the multi-stage non-negative matrix factorization model MNMF [5], and the heterogeneous graph attention network model HGAT [6]. However, there have been few surveys on HNE. Some studies have reviewed the fields of network and graph embedding, but these studies are not specific to HINs [7–14]. To the best of our knowledge, there are several HNE surveys so far [15–19]. Yang et al. [15] reviewed HNE and categorized HNE models into three categories, including proximity-preserving, message-passing, and relation-learning methods. Xie et al. [16] divided current HNE models into path-based, semantic unit-based, and other methods, and each type was further divided into traditional and deep learning-based methods. In addition, from the perspective of modeling goals, Wang et al. [18] categorized existing HNE models into four categories: structure-preserved, attribute-assisted, application-oriented, and dynamic heterogeneous graph embedding models. Dong et al. [17] and Ji et al. [19] also briefly reviewed HNE.

However, these existing surveys on HNE either lack clear classification patterns, or summarize model types in a manner that is not comprehensive and lack in-depth comparison and analysis. To bridge the gap, we provide a comprehensive survey of the state-of-the-art HNE research in this paper. Specifically, we first define a classification mode for HNE study based on an encoder-decoder framework, and explore the major components of an HNE model under the framework. Then, we present a systematic and comprehensive survey of the six categories in the taxonomy. We analyze the basic characteristics, modeling capabilities, advantages and disadvantages of each type of HNE model. Further, we summarize the application areas, publicly available benchmark datasets, open source codes/tools, and give performance comparisons of some typical HNE models on the DBLP dataset for link prediction tasks. Finally, we discuss the open issues and suggest future research directions. The unique contributions of this paper can be summarized as follows:

- **From a technical perspective, we use a more primitive, fundamental, and systematic classification mode for HNE approaches**. Unlike most surveys that classify models from the perspective of modeling goals, we use the encoder-decoder framework to classify the existing HNE models from a technical perspective. This classification mode is more primitive, fundamental, and systematic. It can explicitly capture the methodological diversity and place the various approaches on an equal symbolic and conceptual basis. We elaborate on major components that an HNE model usually contains under the encoder-decoder framework, including encoder, decoder, empirical proximity matrix, and loss function.
- **We provide a comprehensive survey of HNE research**. Based on the proposed classification mode, existing HNE methods can be mainly divided into six categories: matrix factorization (MF), random walk (RW), AutoEncoder (AE), graph neural network (GNN), knowledge graph embedding (KGE), and hybrid (HB) methods. We provide a systematic and comprehensive survey of each type of HNE model. For each model type, we first overview its overall common characteristics and modeling ideas. Then, taking the modeling goals and capabilities of the representative HNE models as the main clues, we conduct a systematic and comprehensive overview of each model type. We use extensive tables to analyze and demonstrate the uniqueness of each representative HNE model in the definition of encoder, decoder, loss function, and empirical proximity matrix. We also analyze the modeling capabilities of them and highlight their novel contributions. Finally, we summarize the overall strengths and weaknesses (or challenges) of each type of HNE model to uncover more potentially competitive HNE model frameworks. We believe that these in-depth and extensive analyses and summaries can be helpful in guiding the development of future novel HNE models and aid researchers and practitioners in choosing appropriate HNE frameworks for specific ML tasks.
- **We provide a wealth of valuable relevant resources**. We summarize the HNE-related application fields, publicly available benchmark datasets, open source codes, and tools, which are rich resources for researchers and practitioners in this field. More over, we provide performance comparisons of some typical HNE models on the DBLP dataset for the link prediction task.
- **We suggest nine promising research directions in the field of HNE**. We discuss open and challenging issues and propose nine promising research directions in the field of HNE in terms of aspects such as interpretability, scalability, heterogeneity, multi-objectiveness, and robustness. For each direction, we provide an in-depth analysis of the inadequacies in the current research and explore future research directions.

The remainder of this survey paper is organized as follows. Section 2 defines the research problem and elaborates our classification mode. Section 3 provides a comprehensive survey of the

state-of-the-art HNE research. Section 4 summarizes the application fields, benchmark datasets, open source codes/tools, and performance evaluations. Section 5 discusses open issues and outlines potential future research directions. Section 6 presents concluding remarks.

2 Research Problem

We use uppercase boldface letters for matrices and lowercase boldface letters for vectors. Table 1 lists the key symbols used in this paper.

Table 1: Symbol description

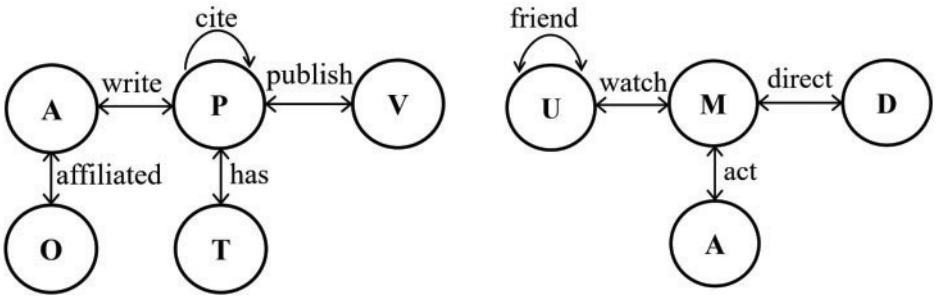
Symbol	Description	Symbol	Description
$\mathcal{G}$	An HIN	$\mathcal{R}$	Set of relation types
$\mathcal{E}$	Set of edges	$\mathcal{T}$	Set of node types
τ	Node type mapping function	φ	Edge type mapping function
$\boldsymbol{A}$	Adjacency matrix of $\mathcal{G}$	$\mathcal{X}$	Set of node attributes
e_{ij}	One edge $(v_i, v_j) \in \mathcal{E}$	w_{ij}	Edge weight of e_{ij}
$ENC(\cdot)$	Encoding function	$DEC(\cdot)$	Decoding function
$\mathcal{L}$	Loss function defined on $\mathcal{G}$	$\hat{\boldsymbol{S}}$	Empirical proximity matrix
$\mathcal{G}^{(k)}$	kth subnetwork of $\mathcal{G}$	$\mathcal{V}^{(k)}$	Node set of $\mathcal{G}^{(k)}$
$\mathcal{E}^{(k)}$	Edge set of $\mathcal{G}^{(k)}$	$\hat{\boldsymbol{S}}^{(k)}$	Empirical matrix of $\mathcal{G}^{(k)}$
$deg(v_i)$	Degree of node v_i	$\mathcal{L}^{(k)}$	Loss function defined on $\mathcal{G}^{(k)}$
$V_l \in \mathcal{T}$	A node type in $\mathcal{G}$	$\mathcal{V}_l$	Set of nodes of type V_l
ϕ	A meta-path	$\boldsymbol{\Phi}$	Set of meta-paths
$\boldsymbol{D}$	Degree matrix of $\mathcal{G}$ (where $\boldsymbol{D}_{ii} = \sum_j \boldsymbol{A}_{ij}$)	K	Numbers of subnetworks in $\mathcal{G}$
$\boldsymbol{Z}$	Node embedding matrix of $\mathcal{G}$	$\boldsymbol{z}_v$	Embedding of node $v \in \mathcal{V}$
$\boldsymbol{z}_v^{(k)}$	Node embedding vector for $v \in \mathcal{G}^{(k)}$	$AGG^S(\cdot)$	Semantic aggregation function
$AGG^N(\cdot)$	Neighbor aggregation function	σ	Activation function
$\mathcal{N}(v)$	Set of neighbors of node v	$\mathcal{N}_l(v)$	Set of neighbors of node v of type V_l
$< h, r, t >$	A triple in knowledge graph	$f_r(h, t)$	Scoring function for $< h, r, t >$
$\mathcal{V}_s$	Set of nodes with attributes	$\mathcal{L}_Y$	Loss for a specific ML task Y
$\mathcal{L}_{Reg}$	Regularization term loss	$\mathcal{L}_{Rec}$	Loss for a recommender system
Δ^+	Set of positive data samples	Δ^-	Set of negative data samples
$\mathcal{Y}$	Set of labeled data	$\boldsymbol{a} \odot \boldsymbol{b}$	Element-wise product
$\boldsymbol{a} \otimes \boldsymbol{b}$	Outer product	$\boldsymbol{a} \cdot \boldsymbol{b}$	Inner product
$\boldsymbol{a} \oplus \boldsymbol{b}$	Concatenate two vectors	$[\boldsymbol{a}; \boldsymbol{b}]$	Concatenate two vectors

2.1 *Heterogeneous Information Network*

Definition 1. **Heterogeneous Information Network** (HIN) [4]: An HIN can be defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X}, \tau, \varphi, \mathcal{T}, \mathcal{R})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges of the network $\mathcal{G}$. $\mathcal{T} = \{V_1, V_2, ..., V_{|\mathcal{T}|}\}$ is the set of node types of $\mathcal{G}$, and $\mathcal{R} = \{r_1, r_2, ..., r_{|\mathcal{R}|}\}$ is the set of relation

types of $\mathcal{G}$. Each node $v \in \mathcal{V}$ has a node type, represented by a mapping function $\tau(v)$: $\mathcal{V} \rightarrow \mathcal{T}$. Each edge $e \in \mathcal{E}$ has an edge type, represented by a mapping function $\varphi(e) : \mathcal{E} \rightarrow \mathcal{R}$. $\mathcal{X}$ is the set of node attributes in $\mathcal{G}$.

Fig. 1 presents two example HINs. The bibliographic network in Fig. 1a is a typical HIN that contains five types of nodes: author (A), paper (P), journal (V), organization (O), and topic (T). A P-type node has different types of links: P-A (a paper is written by an author), P-P (a paper cites another paper), P-V (a paper is published at a venue), among others. The Douban Movie network in Fig. 1b is also an HIN, and it contains four types of entities: user (U), movie (M), director (D), and actor (A).



(a) A bibliography network (b) Douban movie network

Figure 1: Example Heterogeneous Information Networks (HINs)

According to the different types of semantic relations, a heterogeneous network $\mathcal{G}$ can usually be divided into multiple different subnets: $\mathcal{G} = \mathcal{G}^{(1)} \cup \mathcal{G}^{(2)} \cup ... \cup \mathcal{G}^{(K)}$, $K = |\mathcal{R}|$. Among them, $\mathcal{G}^{(k)} = \left\{\mathcal{V}^{(k)}, \mathcal{E}^{(k)}, \mathcal{X}^{(k)}, \tau, \varphi\right\}$ is the kth subnetwork of $\mathcal{G}$, which can be a homogeneous or bipartite, directed or undirected network. $\boldsymbol{A}^{(k)}$ is the adjacency matrix of $\mathcal{G}^{(k)}$.

2.2 Heterogenous Network Embedding

Definition 2. **Heterogeneous Network Embedding** (HNE): Given an HIN $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X}, \tau, \varphi, \mathcal{T}, \mathcal{R})$, the task of HNE is to learn a mapping function $f : \mathcal{V} \rightarrow \mathbb{R}^d$ that embeds each node $v \in \mathcal{V}$ into a low-dimensional vector $\boldsymbol{z}_v \in \mathbb{R}^d$ with $d \ll |\mathcal{V}|$. The learned node embeddings need to be able to capture the original network information. Generally, the network information includes structural, semantic, attribute, and label information.

Next, we explain the three main modeling goals of general HNE models:

- To preserve structural information (**Preserve ST**): the generated embeddings are able to preserve the topological proximity (low-order proximity such as 1^{st}, 2^{nd} [20,21] or higher-order proximity, where n^{th}, $n > 2$) or the community structure of the HINs.
- To preserve attribute information (**Preserve AT**): the generated embeddings incorporate the affiliated attribute information of nodes or edges in the HINs.
- To preserve semantic information (**Preserve SM**): the model differentiates between different types of semantic relations when generating network embeddings.

Notably, in addition to generating embeddings for nodes in HINs, some studies generate embeddings for edges, subnetworks, or entire networks. These studies are also studies of HNE. However, because edge and subnet embeddings are also based on node embeddings and generating node

embeddings is the main task of most ML models, this survey focuses on the node embedding models and introduces some extended models.

2.3 Overview of Approaches: An Encoder-Decoder Framework

From a technical perspective, we use an encoder-decoder framework depicted in Fig. 2 to classify and review existing state-of-the-art HNE approaches. In this section, we first discuss the methodological components of an HNE model under this framework.

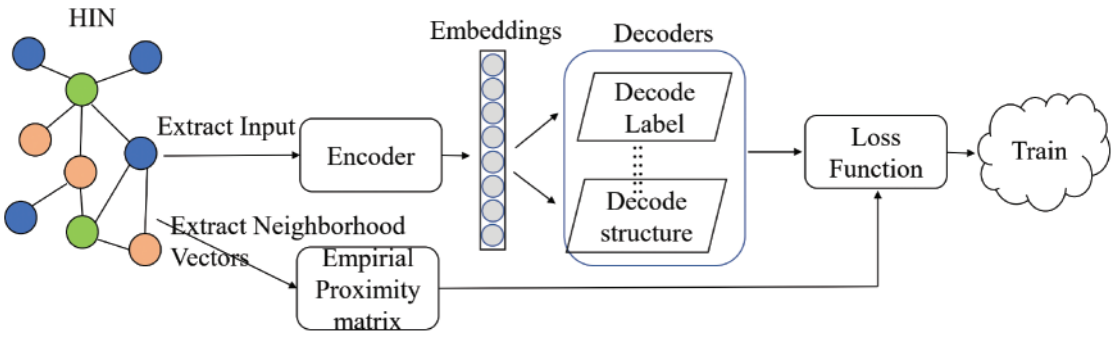


Figure 2: Overview of the encoder-decoder framework

The encoder aims to map each node v in the heterogeneous network $\mathcal{G}$ to a low-dimensional vector space, and the decoders aim to reconstruct the information of the original heterogeneous network based on the learned low-dimensional feature representations. The internal logic is that, if the model can reconstruct the graph structure and semantic relations of the original network from the encoded embeddings, then the learned embeddings should contain all information necessary for downstream ML tasks [8]. Formally, the encoder of an HNE model is a function,

$$ENC : \mathcal{V} \rightarrow \mathbb{R}^d, d \ll |\mathcal{V}|, \tag{1}$$

maps nodes to low dimensional vecotor embeddings $\boldsymbol{z}_v \in \mathbb{R}^d$. As mentioned earlier, a heterogeneous network $\mathcal{G}$ often contains several different subnetworks $\mathcal{G}^{(1)}$, $\mathcal{G}^{(2)}$,..., $\mathcal{G}^{(K)}$. If an HNE model initially maps each node v in each subnetwork $\mathcal{G}^{(k)}$ into different vector spaces, then it usually uses a semantic aggregation function $AGG^S(\cdot)$[1] to aggregate the node embeddings of multiple subnets to generate a unified node embedding, i.e., $\boldsymbol{z}_v = \underset{k}{AGG^S}(\boldsymbol{z}_v^{(k)}), k = 1, 2, ..., K$.

A decoder is also a function that tries to reconstruct the information of the original HIN from the learned node embeddings. To preserve the structural information of the HIN, many HNE models define a pairwise decoder as

$$DEC : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R} \tag{2}$$

to map the pair of node embeddings $\boldsymbol{z}_i, \boldsymbol{z}_j$ to a real-valued structural proximity number which measures the proximity of the two nodes v_i and v_j. If the empirical proximity of v_i and v_j is represented by $\hat{s}_{ij}$, the objection function of the HNE model is usually defined as the sum of the distances between the decoded and empirical proximity of each node pair in the dataset:

[1]In GNN models, a neighbor aggregation function $AGG^N(\cdot)$ is usually used to aggregate the feature of neighbors around each node. Note that in this section of the survey, the aggregation function $AGG^S(\cdot)$ specifically refers to an aggregation of the features for different semantics of the same node. This is semantic-level aggregation, which must be distinguished from node-level feature aggregation in GNN models.

$$\mathscr{L} = \sum_{(v_i,v_j)\in\Delta} \mathscr{D}(s_{ij}, \hat{s}_{ij}), \tag{3}$$

where $\mathscr{D}(\cdot)$ is the distance function. For an HIN, the loss is often the sum of the losses of each subnetwork, i.e., $\mathscr{L} = \sum_{k=1}^{K} \mathscr{L}^k$. After the HNE model has been trained, the node embeddings can be obtained. They can be used for various downstream ML tasks, such as node classification and recommender systems.

In summary, in the encoder-decoder framework, an HNE model typically comprises four components:

- An empirical proximity matrix $\hat{\boldsymbol{S}} : \mathscr{V} \times \mathscr{V} \to \mathbb{R}$. The matrix $\hat{\boldsymbol{S}}$ defines the empirical similarity between each node pair in the HIN. Many models define the empirical proximity matrix as the adjacency matrix $\boldsymbol{A}$, the nth power of the adjacency matrix $\boldsymbol{A}^n$, or the positive pointwise mutual information (PPMI) matrix [22], etc. The definition of $\hat{\boldsymbol{S}}$ largely reflects whether an HNE model considers low-order or high-order structural information when modeling network topology information.
- An encoding function $ENC(\cdot)$. This function maps the model inputs into node embeddings. Depending on the heterogeneous network, the model input may vary. Common model inputs include one-hot identification vectors of nodes, node attribute vectors, or adjacency vectors. The encoder function usually contains many trainable model parameters, which can be very simple, such as a simple linear transformation, or very complex, such as a deep neural network structure. If an HNE model initially maps nodes in each different subnet to a different vector space, then it usually uses a semantic aggregation function $AGG^S(\cdot)$ to aggregate the node embeddings in different subnets to generate the final unified node representations.
- A decoding function $DEC(\cdot)$. A decoder of an HNE model usually contains one or more decoding functions that attempt to reconstruct the information of the original HIN. Most HNE models use pairwise decoding functions when modeling the structural information, and the most common form is the inner product, i.e., $DEC(\boldsymbol{z}_i, \boldsymbol{z}_j) = \sigma(\boldsymbol{z}_i \cdot \boldsymbol{z}_j)$. Models based on KGE always use a ternary scoring function.
- A loss function $\mathscr{L}$. The loss function is the optimization objective of an HNE model. For many unsupervised HNE models, the optimization goal is to minimize the distance between the reconstructed and empirical proximities. To reduce model complexity, most deep learning-based models add a regularization term $\mathscr{L}_{Reg}$ to $\mathscr{L}$. Some HNE models incorporate the loss $\mathscr{L}_Y$ of subsequent ML tasks into $\mathscr{L}$ as information for supervised learning. As such models tipically use part of the label information when generating node embeddings, they are semi-supervised approaches.

3 Heterogenous Network Embedding: State-of-the-Art Approaches

Based on the encoder-decoder framework, existing HNE models can be divided into six categories: MF, RW, AE, GNN, KGE, and Hybrid. As explained later, there are distinct differences in the major components of each model type. In Fig. 3, we present the proposed taxonomy for summarizing HNE techniques.

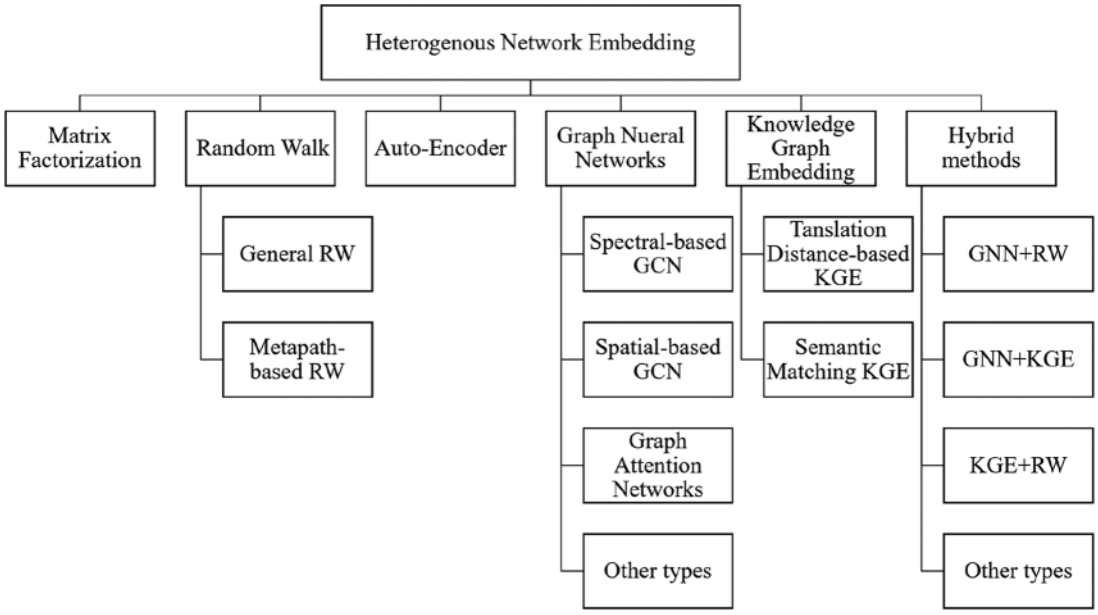


Figure 3: The proposed taxonomy for summarizing Heterogenous Network Embedding (HNE) techniques

3.1 Matrix Factorization-Based Models

Matrix factorization (MF) is an early feature representation learning method. Many MF-based homogeneous network embedding models have been proposed, such as Graph Factorization [23], Laplacian Eigenmaps [24], and HOPE [25]. The encoder of an MF-based model is usually a direct encoding, i.e., the product of a node embedding matrix $\boldsymbol{Z} \in \mathbb{R}^{d \times |\mathcal{V}|}$ and a one-hot vector $\boldsymbol{v}_i$ that identifies each node v_i:

$$ENC(v_i) = \boldsymbol{Z} \cdot \boldsymbol{v}_i. \tag{4}$$

In this case, the encoding function is a simple "embedding lookup", and the embedding matrix $\boldsymbol{Z}$ is directly optimized. The decoder of an MF model is usually defined as the inner product of the two node embeddings as follows:

$$DEC(\boldsymbol{z}_i, \boldsymbol{z}_j) = \boldsymbol{z}_i \cdot \boldsymbol{z}_j. \tag{5}$$

Thus, if the loss function is defined as $\mathcal{L} = \sum_{(i,j)\in\mathcal{E}} \left\| \hat{\boldsymbol{s}}_{ij} - \boldsymbol{z}_i \cdot \boldsymbol{z}_j \right\|_2^2$, then the optimization goal of the HNE model is approximately equivalent to factorizing the empirical similarity matrix, i.e., $\hat{\boldsymbol{S}} \approx \boldsymbol{Z}^T \boldsymbol{Z}$. In addition, although some models do not use an inner-product decoder (e.g., Laplacian Eigenmaps [24] uses $\left\| \boldsymbol{z}_i - \boldsymbol{z}_j \right\|^2$) or add nonlinear transformations to the inner-product operation (e.g., LINE [20] uses $\sigma(\boldsymbol{z}_i \cdot \boldsymbol{z}_j)$), after the objective function is deformed, these models are essentially equivalent to decomposing first- or n-th order proximity matrices [8]. Hence, they can still be classified as MF models.

Different from homogeneous network embedding, HNE models need to deal with network heterogeneity. In general, the most common idea of extending a homogeneous MF model to an HIN is to 1) use a specific MF model to model the relationships in different subnetworks separately, and 2)

sum the losses defined on multiple subnetworks and train them together in an HNE model to get the feature representation of each node in the HIN.

PTE [26] is a semi-supervised model for representation learning of heterogeneous text data. Specifically, to improve the efficiency of text embedding and use the label information in the dataset, PTE represents the text co-occurrence and partial label information in the *corpus* as an HIN that contains the following subnetworks: the *word–word* network $\mathcal{G}^{<ww>}$, *word–document* network $\mathcal{G}^{<wd>}$, and *word–label* network $\mathcal{G}^{<wl>}$. Then, it uses the LINE model [20] to model the relationships in each subnetwork. For each subnetwork $\mathcal{G}^{(k)}$, PTE defines the conditional probability of generating an immediate neighbor node v_j from node v_i as follows:

$$p(v_j|v_i) = \frac{exp(\boldsymbol{z}_i \cdot \boldsymbol{z}_j)}{\sum_{v_{j'} \in \mathcal{V}^{(k)}} exp(\boldsymbol{z}_i \cdot \boldsymbol{z}_{j'})}. \tag{6}$$

To preserve the second-order proximity, the loss function for $\mathcal{G}^{(k)}$ is defined as

$$\mathcal{L}^k = -\sum_{v_i \in \mathcal{V}^k} \lambda_i KL(\hat{p}(\cdot|v_i)||p(\cdot|v_i)), \tag{7}$$

where $\lambda_i = deg(v_i)$, and $\hat{p}(v_j|v_i) = \frac{w_{ij}}{deg(v_i)}$ is the empirical proximity. Finally, PTE provides two optional training solutions: 1) jointly training three subnetworks, where the loss is

$$\mathcal{L} = -\sum_k \sum_{(v_i,v_j) \in \mathcal{E}^k} w_{ij} \log p(v_j|v_i), \tag{8}$$

2) first using the unlabeled networks ($\mathcal{G}^{<ww>}$ and $\mathcal{G}^{<wd>}$) to pre-train, and then using the labeled network ($\mathcal{G}^{<wl>}$) for fine-tuning.

In addition to PTE, HRec [27], ISE [28], and LHNE [29] all use an inner-product decoder as PTE. These models are all able to capture the low-order structural features of the HINs, but they share common defects: 1) These models map the node embeddings in different subnetworks to the same vector space, without considering the diverse semantic information contained in different subnetworks; 2) When the losses of multiple subnetworks are fused together in a single model, these HNE models assign equal weights to different subnetworks. However, the densities of different subnetworks in an HIN are usually quite different. Giving equal weights can easily lead to skew problems, i.e., the HNE model converges in some denser subnetworks but does not in less dense subnetworks.

Inspired by the attention mechanism in neural machine translation in recent years, Qu et al. proposed an MVE model [30] for multi-view heterogeneous networks, which solves the problem of automatically learning the different weights of different views (subnetworks). MVE first maps each node in each subnetwork to a different vector space, and then it also uses the LINE model to obtain the node embedding $\boldsymbol{z}_i^{(k)}$ for each node $v_i \in \mathcal{G}^{(k)}$ (see Eq. (7) for the loss definition). To learn robust node representations across different views, MVE uses an attention mechanism to aggregate the view-specific node embeddings for each node as follows:

$$\boldsymbol{z}_i = \sum_{k=1}^{K} \lambda_i^k \boldsymbol{z}_i^k, \tag{9}$$

where λ_i^k is the weight of the kth subnetwork, defined as $\lambda_i^k = \frac{exp(\boldsymbol{q}^k \cdot \boldsymbol{z}_i^C)}{\sum_{k'=1}^{K} exp(\boldsymbol{q}^{k'} \cdot \boldsymbol{z}_i^C)}$. Next, the model defines a regularization term to learn different weights for different views:

$$\mathscr{L}_{Reg} = \sum_{i=1}^{|\mathcal{V}|} \sum_{k=1}^{K} \lambda_i^k \left\| \boldsymbol{z}_i^k - \boldsymbol{z}_i \right\|_2^2. \tag{10}$$

The final loss of the MVE model is the sum of the model loss for multiple subnets, the regularization term, and the loss for a downstream ML task.

However, the MF-based HNE models mentioned above can only capture low-order structural information of the HINs; they cannot do much for the higher-order structural information. Since the higher-order structural information of the network is equally important in many ML applications, some matrix factorization-based models capture the higher-order structural features by changing the definition of the empirical proximity matrix. A^2CMHNE [31], CMF [22], MIFHNE [32] and MNMF [5] models are typical representatives of this approach. The main differences between them are: the A^2CMHNE and MIFHNE models use the metapath-based similarity matrix (MPSM) as the empirical proximity matrix; the CMF model uses the PPMI matrix [3]; and the MNMF model uses a non-negative DeepWalk matrix [33]. Note that, different from the traditional one-round non-negative matrix factorization, the MNMF model iteratively decomposes the residual matrices to gradually reduce the approximation error based on the idea of gradient boosting. The multi-stage matrix factorization framework of MNMF model is shown in Fig. 4. The objective function of the lth stage is defined as

$$\begin{aligned} &\min \sum_{k=1}^{K} (\alpha_k^l)^{\gamma} \left\| \boldsymbol{R}_k^l - \boldsymbol{U}^l \boldsymbol{V}_k^l \right\|_F^2 \\ &s.t. \boldsymbol{U}^l \geq 0, \boldsymbol{V}_k^l \geq 0, \sum_{k=1}^{K} \alpha_k^l = 1, \alpha_k^l \geq 0, \end{aligned} \tag{11}$$

where the residual matrix $\boldsymbol{R}_k^l$ to be decomposed in l-th stage is defined as

$$\boldsymbol{R}_k^l = \begin{cases} \boldsymbol{M}^k, & l = 1 \\ \max(\boldsymbol{R}_k^{l-1} - \boldsymbol{U}^{l-1} \boldsymbol{V}_k^{l-1}, 0), & 2 \leq l \leq q. \end{cases} \tag{12}$$

where $\boldsymbol{M} = \log\left(max\left(\frac{vol(\mathcal{G})}{bT}\left(\sum_{t=1}^{T} \boldsymbol{P}^t\right)\boldsymbol{D}^{-1}, 1\right)\right)$ is the DeepWalk matrix. Since the matrix $\boldsymbol{V}_k^l$ is different in different subnetworks, the node embeddings generated by MNMF contain both common embeddings across different subnetworks and relation-specific node embeddings.

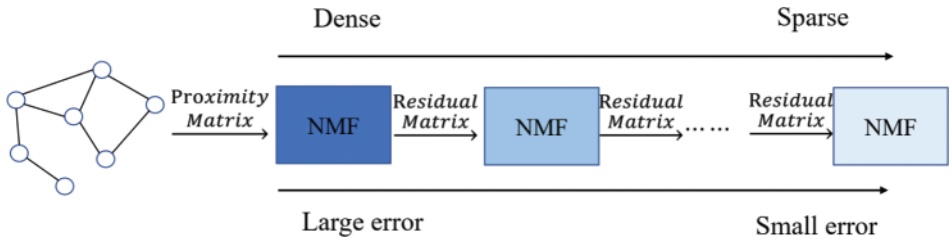


Figure 4: Multi-stage matrix factorization

Table 2: Matrix factorization based models

Model	$\hat{\boldsymbol{S}}$	$AGG^s(\cdot)$	$DEC(\cdot)$	Loss function	Preserve ST	Preserve AT
PTE [26]	$\boldsymbol{A}$	/	$softmax(\boldsymbol{z}_i \cdot \boldsymbol{c}_j)$	Eq. (8)	2^{nd}	no
Hrec [27]	$\boldsymbol{A}$	/	$softmax(\boldsymbol{z}_i \cdot \boldsymbol{c}_j)$	Eq. (8)	2^{nd}	no
ISE [28]	$\boldsymbol{A}$	/	$softmax(\boldsymbol{z}_i \cdot \boldsymbol{c}_j)$	Eq. (8)	2^{nd}	no
LHNE [29]	$\boldsymbol{A}$	/	$softmax(\boldsymbol{z}_i \cdot \boldsymbol{c}_j)$	Eq. (8)	2^{nd}	no
HHE [34]	$\boldsymbol{D}^{-1}\boldsymbol{A}$	/	$\left\|\boldsymbol{z}_i - \boldsymbol{z}_j\right\|^2$	$\sum_{k=1}^{K} \lambda_k \sum_{(i,j)\in\mathscr{E}^k} \hat{s}_{ij} DEC(\boldsymbol{z}_i, \boldsymbol{z}_j)$	1^{st}	no
MVE [30]	$\boldsymbol{A}$	Attention	$softmax(\boldsymbol{z}_i^k \cdot \boldsymbol{c}_j)$	$\sum_{k=1}^{K} \mathscr{L}^{(k)} + \eta \mathscr{L}_{Reg} + \sum_{v_i\in\mathscr{Y}} \mathscr{L}_Y$	2^{nd}	no
A^2CM-HNE [31]	MPSM	Attention	$softmax(\boldsymbol{z}_i \cdot \boldsymbol{c}_j)$	$-\sum_{\phi\in\Phi} \sum_{v_i\in\mathscr{V}} \sum_{v_j\in\mathscr{N}^{\phi}(v_i)} \hat{s}_{ij} \log DEC(\boldsymbol{z}_i, \boldsymbol{z}_j) + \mathscr{L}_Y$	n^{th}	yes
MEGA [35]	GraphSim Matrix	/	$\boldsymbol{z}_i \cdot \boldsymbol{z}_j$	$\alpha \left\|\hat{\boldsymbol{S}} - \boldsymbol{Z}\boldsymbol{Z}^T\right\|_F^2 + \|\mathscr{S} - [Z, Z, W]\|_F^2$	n^{th}	no
CMF [22]	PPMI Matrix	/	$\boldsymbol{z}_i \cdot \boldsymbol{z}_j$	$\sum_{k=1}^{K} \sum_{(i,j)\in\mathscr{E}^k} (DEC(\boldsymbol{z}_i, \boldsymbol{z}_j) - \hat{s}_{ij})^2 + \mathscr{L}_{Reg}$	$1^{st} \sim n^{th}$	no
MNMF [5]	DeepWalk Matrix	CONCAT	$\boldsymbol{z}_i \cdot \boldsymbol{c}_j$	$\sum_{k=1}^{K} (\alpha_k^l)^{\gamma} \left\|\boldsymbol{R}_k^l - \boldsymbol{U}^l \boldsymbol{V}_k^l\right\|_F^2$	$1^{st} \sim n^{th}$	no
SAHE [36]	Aggregated MPSM	/	$\sigma(\boldsymbol{z}_i, \boldsymbol{z}_j)$	$\sum_{(v_i,v_j)\in\Delta} KL(\hat{s}_{ij} \| DEC(\boldsymbol{z}_i, \boldsymbol{z}_j))$	n^{th}	no
MIFHNE [32]	MPSM	/	$\boldsymbol{z}_i \cdot \boldsymbol{c}_j$	$\|\hat{\boldsymbol{S}} - \boldsymbol{U}\boldsymbol{Z}^T\|_F^2 + \alpha\|\boldsymbol{Y} - \boldsymbol{V}\boldsymbol{Z}^T\|_F^2 + \beta\|\boldsymbol{X} - \boldsymbol{W}\boldsymbol{Z}^T\|_F^2$	$1^{st} \sim n^{th}$	yes

Table 2 provides an overview of some typical existing MF-based HNE models. In summary, MF-based HNE models are mainly convenient for modeling the structural information of the HINs. They usually have relatively simple encoding and decoding functions and run efficiently. However, this type of model usually has the following disadvantages:

- Such models have poor ability to capture the attribute information and diverse semantic information of heterogeneous networks. Although some MF models take attribute information into account when generating node embeddings (such as MIFHNE and A^2CMHNE), they usually require separate encoding of attribute information.
- In some MF models that consider higher-order proximity (such as MEGA [35], CMF [22], and MNMF [5]), the proximity matrices to be decomposed are usually dense matrices. For a large-scale network, storing and decomposing a large dense proximity matrix consumes significant memory resources, which makes it difficult to implement on ordinary computing platforms.
- Most encoders of such models are direct encoding, and the model input depends on the number of nodes; therefore, they are generally transductive models, which are difficult to extend to dynamic networks.

3.2 Random Walk-Based Models

Random walk (RW) based models are also very common network embedding models. Typical homogeneous random walk models are DeepWalk [33] and node2vec [37]. In contrast to MF-based models, the neighbor nodes in RW-based models are defined as nodes that co-occur in a short sequence of random walks. The optimization goal of most RW models is to make nodes that frequently co-occur in short sequences of random walks on heterogeneous networks have similar embeddings.

That is, this kind of approach is to learn embeddings so that

$$DEC(\boldsymbol{z}_i, \boldsymbol{z}_j) = \frac{exp(\boldsymbol{z}_i \cdot \boldsymbol{z}_j)}{\sum_{v \in \mathcal{V}} exp(\boldsymbol{z}_i \cdot \boldsymbol{z}_v)} \approx p_{\mathcal{G}}(v_j|v_i), \tag{13}$$

where $p_{\mathcal{G}}(v_j|v_i)$ denotes the conditional probability of randomly walking from node v_i to node v_j in the HIN.

Compared with MF models, random walk-based models have better flexibility and can easily model higher-order neighbor relationships. Furthermore, if the random walk path-sampling process is given some strategic control, such as restricting it to follow a specific relation path (meta-path), it is convenient for a random walk model to capture the specific semantic information contained in the HINs.

Random walk-based HNE models are generally implemented in two stages: 1) sampling a large number of random walk sequences in the HIN from each node according to a specific strategy, 2) learning the optimized node embeddings using the skip-gram (or CBOW) model.

3.2.1 General Heterogeneous Random Walk Embedding Model

In the context of heterogeneous graphs containing nodes from different domains, classical random walks are biased to highly visible domains where nodes are associated with a dominant number of paths [4]. To overcome the skewness problem that random walk may cause, Hussein et al. proposed a JUST model [38] that uses a special *Jump & stay* random walk strategy. JUST can effectively control when the walker performs a random walk, the next step is to choose to walk to a homogeneous node (Stay) or a heterogeneous node (Jump). If the next step is to choose Stay, then the model uses a queue Q_{hist} of length m to store the sequence of node types sampled in the past m steps. In this way, the model avoids frequent resampling of the sampled node types by artificially controlling the type of sampled nodes at each step, overcoming the skewness problem to a certain extent. Similar to JUST, the MARU model [39] is also a RW-based HNE model aimed at the skewness problem. However, unlike JUST, MARU defines a bidirectional extended random walk strategy, and allows each node to have different representations in different contexts, thus enhancing the semantic modeling capability of the model.

Heterogeneous networks contain rich semantic relationships. If the HNE models do not distinguish between different relationships when modeling, the generated node representations are bound to lose rich semantic information, such as HINE [40] and JUST. To deal with the various semantic relations contained in HINs, the MNE model [41] divides the embedding of each node in the subnetwork $\mathcal{G}^r$ into two parts: common embedding $\boldsymbol{c}_i \in \mathbb{R}^d$, which is shared across all the relation types, and an embedding $\boldsymbol{u}_i^{(r)} \in \mathbb{R}^s$ oriented toward a specific view r, which is used to capture the distinct property of each sub-network. In this way, the feature representation $\boldsymbol{z}_i^{(r)}$ of a node v_i in a specific subnetwork $\mathcal{G}^r$ can be expressed as

$$\boldsymbol{z}_i^{(r)} = \boldsymbol{c}_i + \alpha^{(r)} W^{(r)} \boldsymbol{u}_i^{(r)}, \tag{14}$$

where $\alpha^{(r)}$ is the weight of relation r and $W^{(r)} \in \mathbb{R}^{d \times s}$ is the r-view-specific transformation matrix. Next, the model samples several fixed-length random walk sequences in each subnetwork $\mathcal{G}^r$ and then uses the skip-gram algorithm to obtain the view-specific node embeddings.

Other general RW-based HNE models are summarized in Table 3. In particular, the hyper-gram [42] model uses a novel indicator to describe the indecomposability of the hyperedge in a heterogeneous hypergraph, followed by a random walk sampling algorithm based on a hyper-path and the corresponding hyper-gram optimization algorithm. The GERM [43] model uses a genetic

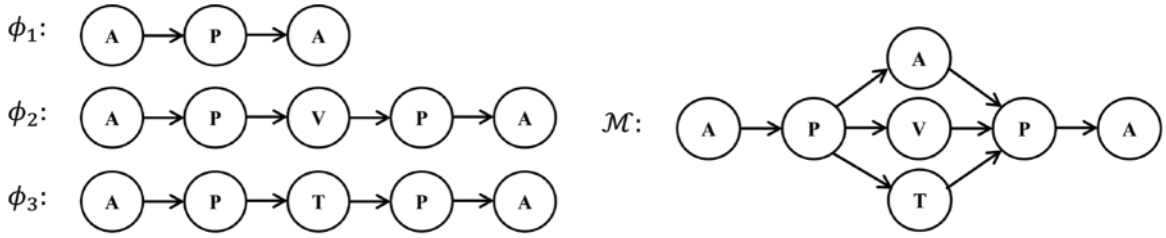
algorithm to select the most informative relation type (subgraph pattern) in a heterogeneous network pattern for a specific ML task, and then uses the generated edge type activation vector (ETAV) to guide the random walk process to reduce noise and model complexity. These two models provide new perspectives for designing random walk sampling procedures. Moreover, because GERM mines the most informative relational patterns for specific ML tasks in the HIN schema, it is also intrinsically related to the automatic mining of meta-paths to be discussed later.

3.2.2 Metapath-Based Random Walk Embedding Models

Since the concept of meta-path was proposed by Sun et al. [44] in 2011, numerous studies have used metapath-based random walk techniques to generate node embeddings. Compared with general RW-based models, metapath-based RW models better capture various types of semantic information contained in the HINs. The characteristic of this type of model is that when a walker performs a random walk in an HIN, it should follow a predefined specific meta-path.

Definition 3. **meta-path** [4]: A meta-path ϕ is a sequence of node types of length L: $\phi : V_1 \xrightarrow{r_1} V_2 \xrightarrow{r_2} \ldots V_l \xrightarrow{r_l} V_{l+1} \ldots \xrightarrow{r_{L-1}} V_L$, where $V_1, V_2, \ldots, V_L \in \mathscr{T}$ are node types of the HIN, and $r_1, r_2, \ldots, r_{L-1} \in \mathscr{R}$ are the meta relationship between two node types. A meta-path $R = r_1 \circ r_2 \circ \ldots \circ r_{L-1}$ defines a composite relationship between two node types V_1 and V_L.

Taking Fig. 1a as an example, the three meta-paths shown in Fig. 5a can be defined as $\phi_1 : A \rightarrow P \rightarrow A$, $\phi_2 : A \rightarrow P \rightarrow V \rightarrow P \rightarrow A$, and $\phi_3 : A \rightarrow P \rightarrow T \rightarrow P \rightarrow A$. These meta-paths describe three distinct relationships between two author-type entities: those co-publishing papers, those publishing papers in the same venues, and those publishing on the same topics.



(a) Three different meta-paths ϕ_1,ϕ_2 and ϕ_3 (b) A meta-graph $\mathscr{M}$

Figure 5: Examples of meta-paths and a meta-graph based on Fig. 1

To address the network heterogeneity challenge, Dong et al. proposed a classic metapath-based random walk model called metapath2vec [4]. The model defines a random walk sampling method based on a meta-path $\phi : V_1 \xrightarrow{r_1} V_2 \xrightarrow{r_2} \ldots V_l \xrightarrow{r_l} V_{l+1} \ldots \xrightarrow{r_{L-1}} V_L$ to generate sequences of sampled nodes. At each step of the random walk, the probability of walking from the current node v_i (assuming its node type is V_l) to the next node v_{i+1} is defined as

$$p(v_{i+1}|v_i^l;\phi) = \begin{cases} \dfrac{1}{|\mathscr{N}_{l+1}(v_i^l)|}, & if\,(v_{i+1}, v_i^l) \in \mathscr{E} and \tau(v_{i+1}) = V_{l+1} \\ 0, & otherwise \end{cases} \tag{15}$$

Then, based on the sampled random walk sequences, the heterogeneous skip-gram algorithm is used to learn the model parameters. The loss function is defined as

$$\mathscr{L} = -\sum_{v_i \in \mathscr{V}} \sum_{V_l \in \mathscr{T}} \sum_{v_j \in \mathscr{N}_{l+1}(v_i)} \left[\log(\boldsymbol{z}_i \cdot \boldsymbol{z}_j) + \sum_{m=1}^{M} \mathbb{E}_{v_{j'} \sim P_n^{l+1}(u)} \log \sigma(-\boldsymbol{z}_i \cdot \boldsymbol{z}_{j'}) \right]. \tag{16}$$

Metapath2vec can effectively deal with the heterogeneity issue, and capture the structural and specific semantic information of the HINs. However, it only uses a single meta-path in random walk sequence sampling, which makes the generated node representations capture only limited semantic information; meanwhile, sequence sampling based on a single meta-path can easily lead to the sampling of numerous short random walk sequences, causing the sparsity problem [45].

To capture more diverse semantic relations in HINs, Zhang et al. proposed a MetaGraph2vec model [45] that defines a meta-graph-based random walk strategy. A meta-graph is essentially a combination of multiple meta-paths, e.g., the meta-graph $\mathscr{M}$ defined in Fig. 5b is the combination of the three meta-paths ϕ_1, ϕ_2, and ϕ_3 in Fig. 5a. When sampling a random walk sequence based on the meta-graph $\mathscr{M}$, if the current random walker passes through a node of type A to a node of type P, then at the next step, it can choose any type of node in $\{A, V, T\}$ to walk. As the meta-graph contains more diverse semantic relations than a single meta-path, the metagraph2vec model can capture richer semantic information than metapath2vec. However, the metagraph2vec model essentially assigns the same weights to different meta-paths, rather than distinguishing them. This makes the resulting node embeddings unable to capture more important relational paths. The HERec [46] and HueRec [47] models handle the weighting of multiple meta-paths well. Both of these two models are random walk-based HNE models designed for recommendation systems. Specifically, HERec first uses the metapath2vec model to learn the node embedding $\boldsymbol{z}_v^{(\phi)} \in \mathbb{R}^d$ of each user node $v \in \mathscr{U}$ and item node $v \in \mathscr{I}$ based on a specific meta-path ϕ, then it uses three different ways to aggregate node feature representations based on multiple meta-paths. Finally, it feeds the aggregated embeddings into the recommendation model to generate rating predictions. Compared with HERec, the HueRec model has some differences: first, HueRec uses the PathSim metric [44] to define the empirical proximity between a user node and an item node; secondly, when decoding the similarity between user node u and item node i, HueRec uses a ternary decoder as follows: $s_{u,i,\phi} = \sum_{q=1}^{d} \boldsymbol{z}_u \odot \boldsymbol{z}_i \odot \boldsymbol{z}_\phi^{\Phi}$. Since the decoder of HueRec distinguishes different semantics of different meta-paths, it has a better ability to capture semantic relations.

In addition to the above models, there are many models that also belong to the metapath-based random walk models [31,48–53]. Noted that the HIN2VEC [49] model decomposes all the sampled random walk sequences based on the HIN schema into short meta-path sequences of length no more than n-hops, and defines a relation type-sensitive ternary function to decode the proximity of the triples. The HeteSpaceyWalk [48] model formalizes the metapath-based random walk process as a higher-order Markov chain. In contrast to all other HNE models that map heterogeneous networks to low-dimensional Euclidean spaces, the HHNE model [52] maps nodes in the HIN into a hyperbolic space. As the random walk-based model is essentially modeling the structural information of the HINs, for the attribute information, the SHNE [50] and A^2CMHNE model [31] design attribute encoders specially and integrate the content embeddings into the HNE models.

Table 3 presents an overview of the characteristics of some typical RW-based HNE models. In essence, the RW-based models also perform matrix factorization [54]. However, unlike MF models, because the neighbor nodes in RW models are neighbors that co-occur in a sequence of random walks, RW-based HNE models always factorize higher-order proximity matrices. Moreover, most RW models use edge sampling to generate data samples. Therefore, the RW-based model avoids the drawback of directly decomposing large and dense matrices, providing better flexibility than MF models. However, this type of model still has some shortcomings:

Table 3: Random walk based models

Model	$AGG^S(\cdot)$	$DEC(\cdot)$	Loss function	Preserve ST	Preserve AT
JUST [38]	/	$\sigma(\boldsymbol{z}_i \cdot \boldsymbol{z}_j)$	$-\sum_{(v_i,v_j)\in\Delta} \hat{s}_{ij} \log DEC(\boldsymbol{z}_i, \boldsymbol{z}_j)$	$1^{st} \sim n^{th}$	no
MNE [41]	/	$softmax(\boldsymbol{z}_i^{(r)} \cdot \boldsymbol{c}_j)$	$-\sum_{r\in\mathcal{R}} \sum_{(i,j)\in\mathcal{E}^{(r)}} \hat{s}_{ij} \log DEC(\boldsymbol{z}_i^{(r)}, \boldsymbol{c}_j)$	$1^{st} \sim n^{th}$	no
HINE [40]	/	$softmax(\boldsymbol{z}_i \cdot \boldsymbol{c}_j)$	$-\sum_k \alpha^k \sum_{(v_i,v_j)\in\mathcal{G}^{(k)}} \hat{s}_{ij}^{(k)} \log DEC(\boldsymbol{z}_i, \boldsymbol{c}_j)$	$1^{st} \sim n^{th}$	yes
Hyper-Gram [42]	/	$softmax(\boldsymbol{z}_i \cdot \boldsymbol{c}_j)$	$-\sum_{(v_i,v_j)\in\Delta} \hat{s}_{ij} \log DEC(\boldsymbol{z}_i, \boldsymbol{z}_j) - \sum_{y\in\Delta^+} [\log S_{tuple}(y) - \sum_{y_n\in\Delta^-} \log S_{tuple}(y_n)]$	$1^{st} \sim n^{th}$	no
MARU [39]	$\sum_m p(m\|v_i)\boldsymbol{z}_i^m$	$softmaxt(\boldsymbol{z}_i^m \cdot \boldsymbol{c}_j^{m_c})$	$-\sum_{v_i\in\mathcal{V}} \sum_{m\in M} \sum_{v_j\in C(v_i,m,m_c)} \log p(v_j\|v_i, m, m_c)$	$1^{st} \sim n^{th}$	no
GERM [43]	/	$\sigma(\boldsymbol{z}_i \cdot \boldsymbol{z}_j)$	$-\sum_{v_i\in\mathcal{V}_{AE}} \sum_{t\in\mathcal{T}_{AE}} \sum_{v_j\in\mathcal{N}^t(v_i)} \hat{s}_{ij} \log DEC(\boldsymbol{z}_i, \boldsymbol{z}_j)$	$1^{st} \sim n^{th}$	no
MetaPath-2Vec [4]	/	$softmax(\boldsymbol{z}_i \cdot \boldsymbol{z}_j)$	Eq. (16)	$1^{st} \sim n^{th}$	no
HERec [46]	$\sigma(\sum_{l=1}^{\|\Phi\|} \alpha_i^{(l)} \cdot \sigma(\boldsymbol{W}^{(l)}\boldsymbol{z}_i^{(l)} + b^{(l)}))$	$softmax(\boldsymbol{z}_i \cdot \boldsymbol{z}_j)$	$\mathcal{L}_{Rec} + \lambda\mathcal{L}_{Reg}$	n^{th}	no
MetaGraph-2Vec [45]	/	$softmax(\boldsymbol{z}_i \cdot \boldsymbol{z}_j)$	Eq. (16)	$1^{st} \sim n^{th}$	no
HeteSpacey-Walk [48]	/	$softmax(\boldsymbol{z}_i \cdot \boldsymbol{c}_j)$	$-\sum_{v_i\in\mathcal{V}} \sum_{t\in\mathcal{T}} \sum_{v_j\in\mathcal{N}_t(v_i)} \hat{s}_{ij} \log DEC(\boldsymbol{z}_i, \boldsymbol{c}_j)$	$1^{st} \sim n^{th}$	no
HueRec [47]	/	$\sum_{q=1}^{d} \boldsymbol{z}_u^U \odot \boldsymbol{z}_i^I \odot \boldsymbol{z}_\phi^\Phi$	$-\frac{1}{\|\Phi\|}\sum_{\phi\in\Phi} \sum_{(u,i,\phi)\in\Delta} w_u^\phi \hat{s}_{u,i,\phi} DEC(\boldsymbol{z}_u^U, \boldsymbol{z}_i^I, \boldsymbol{z}_\phi^\Phi) + \mathcal{L}_{Rec} + \mathcal{L}_{Reg}$	n^{th}	no
HIN2Vec [49]	/	$\sum_{q=1}^{d} \boldsymbol{z}_i \odot \boldsymbol{z}_j \odot \boldsymbol{z}_r$	$\sum_{(i,j,r)\in\Delta} \hat{s}_{ijr} \log DEC(\boldsymbol{z}_i, \boldsymbol{z}_j, \boldsymbol{z}_r)$	$1^{st} \sim n^{th}$	no
SHNE [50]	/	$softmax(\boldsymbol{z}_i \cdot \boldsymbol{z}_j)$	$-\sum_{v_i\in\mathcal{V}} \sum_{t\in\mathcal{T}} \sum_{v_j\in N_t(v_i)} \hat{s}_{ij} \log DEC(\boldsymbol{z}_i, \boldsymbol{z}_j) + \sum_{v\in\mathcal{V}_s} \log p(\boldsymbol{z}_v\|\boldsymbol{x}_v)$	$1^{st} \sim n^{th}$	yes
Esim [51]	/	$\boldsymbol{z}_i \cdot \boldsymbol{z}_j + \boldsymbol{p}_\phi \cdot \boldsymbol{z}_i + \boldsymbol{q}_\phi \cdot \boldsymbol{z}_j + \mu_\phi$	$-\sum_{\rho\in Path^\phi} \sum_{i=1}^{\|\phi\|} \sum_{j=i}^{\|\phi\|} \hat{s}_{ij\rho} \log\sigma(DEC(\boldsymbol{z}_i, \boldsymbol{z}_j))$	$1^{st} \sim n^{th}$	no
HHNE [52]	/	$\sigma(-d_{\mathbb{D}}(\boldsymbol{z}_i, \boldsymbol{z}_j))$	Eq. (16)	$1^{st} \sim n^{th}$	no

- The random walk-based edge sampling can easily lead to skewness issues, which can make the HNE model unbalanced during training; the model is likely to be well-trained in dense subnetworks but far from convergent in sparse subnetworks. Hence, the random walk strategy must be carefully designed to mitigate the skewness problem.
- In random walk models using meta-paths, the definition of the meta-paths generally depends on the engineer's prior knowledge and experience. If the meta-path is not well defined, it may add noise to the HNE model and useful network information may be lost. It would be appreciable to study the automatic extraction of efficient meta-paths from HINs.
- Similar to MF models, most RW models use direct encoding. Because there are no shared parameters between node embeddings, this may affect the training efficiency of the model.
- The random walk models can capture the low- and high-order structural information of the network, but similar to the MF model, they cannot handle the attribute information well: the attribute information usually needs to be encoded separately. For example, in the SHNE model [50], the text attribute information in the HIN is encoded using a specialized gated recurrent unit-based recurrent neural network.

3.3 Autoencoder-Based Models

The AE-based HNE models are different from other types. The model input of this model type is no longer a one-hot identification vector but an adjacency vector $\hat{\boldsymbol{s}}_i \in \mathbb{R}^{|\mathcal{V}|}$ of each node v_i, i.e., the ith row of the empirical proximity matrix $\hat{\boldsymbol{S}}$. The adjacency vectors contain the neighbor structural information of each node in the HIN. The AE-based model uses a nueral network-structured encoder to project the node adjacency vectors into the low-dimensional embedding vectors; and the decoder attempts to recover the input vectors from the learned node embeddings (Fig. 6).

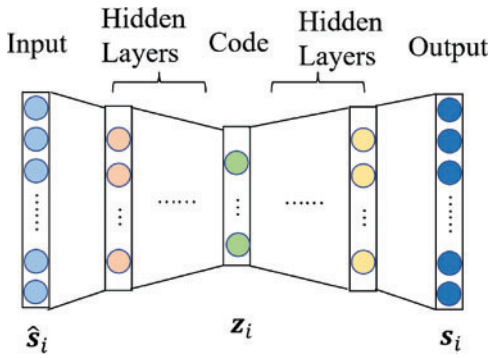


Figure 6: The framework of autoencoder-based models. The encoder of this model type is usually a multi-layer neural network framework (called hidden layers), which maps the input vector $\hat{\boldsymbol{s}}_i$ for node v_i to the embedding vector $\boldsymbol{z}_i$ (code). The decoder is also a multi-layer neural network framework, which attempts to reconstruct the input vectors from the learned code. The optimization goal of the AE-based models is usually to minimize the distance between the input and output vectors, namely: $\mathscr{L} = \sum_{v_i \in \mathcal{V}} \|(\boldsymbol{s}_i - \hat{\boldsymbol{s}}_i) \odot \boldsymbol{b}_i\|_F^2$, where $\boldsymbol{b}_i$ is an indicator vector, which gives more penalty to those non-zero elements

Most AEs are implemented through various types of neural networks, such as feed-forward neural networks (FNNs), sparse AEs, denoising AEs, contractive AEs, and variational AEs. Common AE-based homogeneous network embedding models include SDNE [55], DNGR [56], and VGAE [57]. For HINs, an AE-based HNE model should also fully consider the networks' heterogeneity.

The SHINE model proposed by Wang et al. [58] is a basic heterogeneous extension of homogeneous network embedding. It first uses three AE neural networks to compress user node adjacency vectors in three different subnetworks; then, it aggregates the node representations in different subnetworks to obtain the unified node embeddings. However, because the input to this model is the node's first-order adjacency vector, it can only capture the second-order structural features. DHNE [21] and Event2Vec [59] can capture both first- and second-order structural features. Specifically, to capture the first-order structural information, DHNE uses a deep neural network framework to define a nonlinear multivariate function $S(v_1, v_2, ..., v_n) = MLP(\boldsymbol{z}_1, \boldsymbol{z}_2, ..., \boldsymbol{z}_n)$, and the first-order loss is defined as

$$\mathscr{L}_1 = -\sum_{ijk} \hat{s}_{ijk} \log s_{ijk} - (1 - \hat{s}_{ijk}) \log (1 - s_{ijk}). \tag{17}$$

To capture the second-order structural information, DHNE defines an adjacency matrix $\boldsymbol{A}$ ($= \boldsymbol{H} \cdot \boldsymbol{H}^T - \boldsymbol{D}$) based on the event matrix $\boldsymbol{H}$. Then, the model compresses the matrix $\boldsymbol{A}$ by using the AE framework to obtain the feature representation of the network nodes, and the second-order loss is defined as

$$\mathscr{L}_2 = \sum_{t \in \mathscr{T}} \| sign(\boldsymbol{s}_i^t) \odot (\boldsymbol{s}_i^t - \hat{\boldsymbol{s}}_i^t) \|_F^2. \tag{18}$$

Finally, DHNE merges the two loss functions into one model for unified training.

Besides low-order structural information, the ability of the HNE model to capture high-order structural information is also important. DIME [60] and AMPE [61] extend the definition of neighbor nodes with metapath-based neighbor nodes, thus capturing higher-order structural information of the network. Specifically, DIME first defines multiple meta-paths $\phi_1, \phi_2, ..., \phi_L$ in two HINs $\mathscr{G}^{(1)}$ and $\mathscr{G}^{(2)}$. Then, it calculates the empirical proximity between node pairs $(v_i, v_j) \in \mathscr{G}^{(k)}$ based on each meta-path ϕ_l as follows: $\hat{s}_{ij,\phi_l}^{(k)} = \dfrac{2|\mathscr{P}_{\phi_l}^{(k)}(v_i^{(k)}, v_j^{(k)})|}{|\mathscr{P}_{\phi_l}^{(k)}(v_i^{(k)}, \cdot)| + |\mathscr{P}_{\phi_l}^{(k)}(\cdot, v_j^{(k)})|}$. Next, with all empirical approximation matrices $\hat{\boldsymbol{S}}_{\phi_l}^{(k)}, l = 1, 2, ..., L$ as input, DIME uses a multi-level hybrid AE framework, as shown in Fig. 7, to compress the input into low-dimensional node embeddings. The reconstruction loss for each network $\mathscr{G}^{(k)}$ is defined as

$$\mathscr{L}^{(k)} = \sum_{\phi_l \in \boldsymbol{\Phi}} \sum_{v_i \in \mathscr{V}} \| (\boldsymbol{s}_{i,\phi_l}^{(k)} - \hat{\boldsymbol{s}}_{i,\phi_l}^{(k)}) \odot \boldsymbol{b}_{i,\phi_l}^{(k)} \|_2^2. \tag{19}$$

and the final loss is defined as

$$\mathscr{L} = \mathscr{L}^{(1)} + \mathscr{L}^{(2)} + \alpha * \mathscr{L}^{(1,2)} + \beta * \mathscr{L}_{reg}, \tag{20}$$

As mentioned above, in addition to capturing network structure information, HNE models often need to deal with diverse network attribute information. AMVAE [62] and AEHE [63] are two AE-based HNE models that incorporate content attribute embeddings. For the text information $\mathscr{W}_v = \{w_1, w_2, ..., w_m\}$ and image region information $\mathscr{I}_v = \{i_1, i_2, ..., i_n\}$ contained in each image node $v \in \mathscr{V}$ in the heterogeneous image network, AMVAE first defines an attention model that captures the association between each word and image region. Then, it inputs the text representation sequence corresponding to each image into a long short-term memory model and obtains the content embedding of each image node. Next, it concatenates the content embedding and structure embedding, and inputs the concatenated feature vector into a two-stage hybrid VAE framework to learn the final embeddings

of the image nodes. Owing to the introduction of a multi-level hybrid VAE framework, the robustness of the AMVAE model is enhanced.

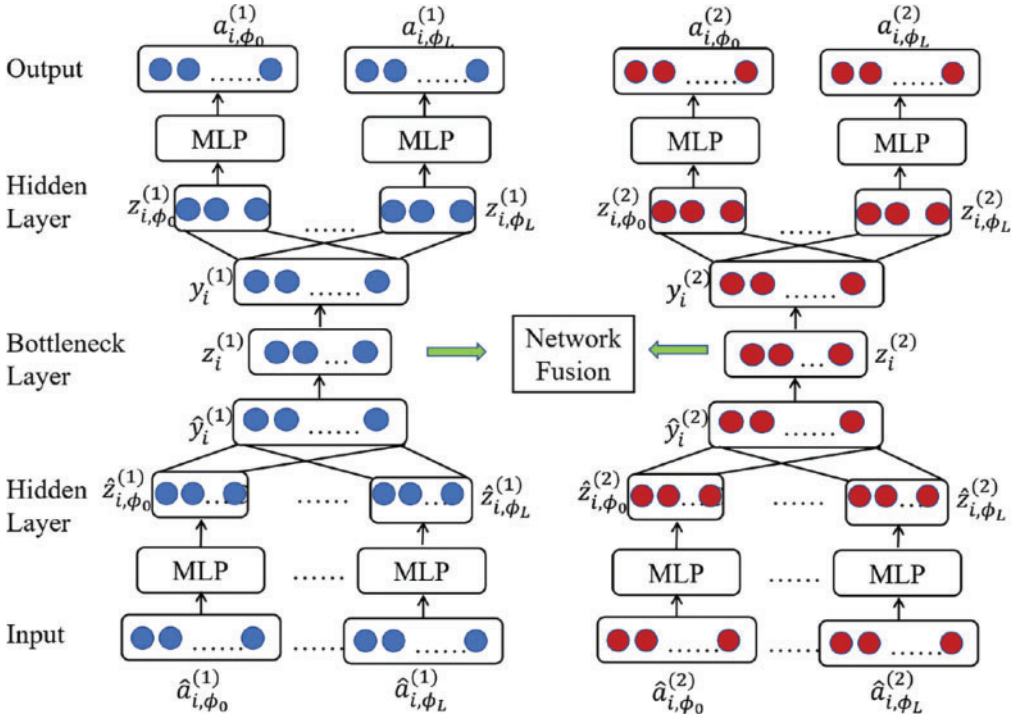


Figure 7: DIME model framework

Table 4 gives an overview of the features of some representative AE-based HNE models. In summary, the overall advantage of an AE-based HNE model is that it can easily compress the neighbor vectors through various types of AE frameworks to capture network structural features directly. However, such models often have the following disadvantages:

- For large- and ultra-large-scale heterogeneous information networks, the input adjacency vectors to AE-based models are usually high-dimensional (tens of thousands to hundreds of millions). Building a general multi-layer AE neural network structure usually involves numerous training parameters in the model. Therefore, the training complexity of such HNE models is very high and difficult to implement on general computing platforms.
- As the input of such models depends on the number of nodes in the network, the models are usually transductive and cannot handle dynamic networks.

Table 4: Autoencoder based models

Model	$\hat{S}$	Encoder	$AGG^S(\cdot)$	Loss Function	Preserve ST	Preserve AT
SHINE [58]	A	MLP	SUM, maxpooling, CONCAT	$\sum_{k=1}^{K}\sum_{v_i\in\mathcal{V}^{(k)}}\alpha_k\left\|(\hat{\boldsymbol{a}}_i^{(k)}-\boldsymbol{a}_i^{(k)})\odot\boldsymbol{b}_i^{(k)}\right\|_2^2+\mathcal{L}_Y+\mathcal{L}_{reg}$	2^{nd}	yes
AMPE [61]	MPSM	MLP	Attention	$\sum_{k=1}^{\lvert\Phi\rvert}\sum_{i\in\mathcal{V}}\alpha_{\phi_k}\left\|(\hat{\boldsymbol{a}}_i^{(\phi_k)}-\boldsymbol{a}_i^{(\phi_k)})\odot\boldsymbol{b}_i^{(\phi_k)}\right\|_F^2+\mathcal{L}_Y$	$1^{st}\sim n^{th}$	no

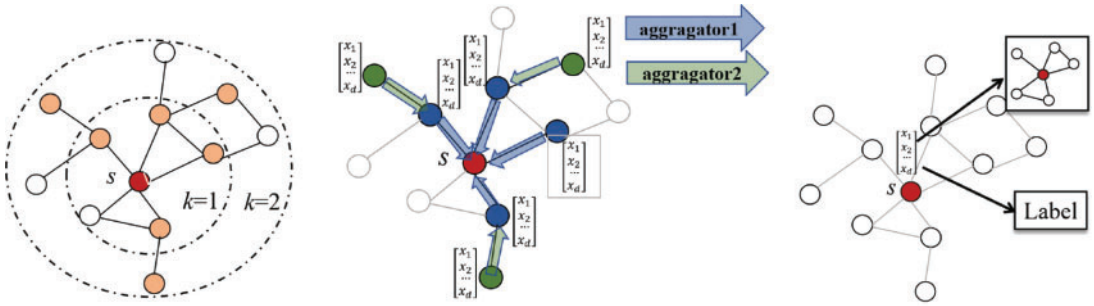
(Continued)

Table 4 (continued)

Model	$\hat{S}$	Encoder	$AGG^S(\cdot)$	Loss Function	Preserve ST	Preserve AT
AMVAE [62]	A,X	VAE	CONCAT	$[\xi - f(g(\mathscr{I}_v, \mathscr{W}_v)) + f(g(\mathscr{I}_v, \bar{\mathscr{W}}_v))]_+ + \sum_{k=1}^{2} [KL(q(\boldsymbol{z}_v^{(k)})\|\boldsymbol{a}_v^{(1)}, \boldsymbol{a}_v^{(2)})\|p(\boldsymbol{z}_v^{(k)})) + H(\boldsymbol{a}_v^{(k)}, \hat{\boldsymbol{a}}_v^{(k)})]$	2^{nd}	yes
DIME [60]	MPSM	MLP	CONCAT	$\sum_{k=1}^{K} \sum_{\phi\in\Phi} \sum_{v_i\in\mathscr{V}^{(k)}} \left\|(\hat{\boldsymbol{a}}_{i,\phi}^{(k)} - \boldsymbol{a}_{i,\phi}^{(k)}) \odot \boldsymbol{b}_{i,\phi}^{(k)}\right\|_2^2 + \left\|\boldsymbol{T}^{(1,2)}\boldsymbol{Z}^{(1)}\boldsymbol{W}^{(1,2)} - \boldsymbol{Z}^{(2)}\right\|_F^2$	$1^{st} \sim n^{th}$	yes
Event-2vec [59]	Event matrix	MLP	AVG	$\sum_{i=1}^{\|\Omega\|} \sum_{t=1}^{\|T_v\|} \left\|(\hat{\boldsymbol{s}}_i^t - \boldsymbol{s}_i^t) \odot \boldsymbol{b}_i^t\right\|_2^2 + \alpha\mathscr{L}_{Reg}$	$1^{st} \sim 2^{nd}$	no
DHNE [21]	$\boldsymbol{H}\boldsymbol{H}^T - \boldsymbol{D}$	MLP	/	$-\sum_{(i,j,k)\in\delta} \hat{s}_{ijk} \log s_{ijk} + (1 - \hat{s}_{ijk}) \log(1 - s_{ijk})$	$1^{st} \sim 2^{nd}$	no
AEHE [63]	MPSM	MLP	/	$\sum_{event} \sum_{i=1}^{m} \left\|\boldsymbol{B}_i \odot (\boldsymbol{S}_i - \hat{\boldsymbol{S}}_i)\right\| + \alpha\mathscr{L}_y + \beta\mathscr{L}_{Reg}$	$2^{nd} \sim n^{th}$	yes

3.4 Graph Neural Network-Based Models

Inspired by the convolutional neural network (CNN), GNNs, which operate on the graph domain, have been developed in recent years [64,65]. GNNs are able to capture dependencies contained in graphs from graph structural information through information propagation (see Fig. 8). Different from previous types of models, the input of a GNN-based HNE model is usually the nodes' affiliate attribute vectors, and its encoding function is a multilayer GNN, which continuously aggregates the features of the neighbor nodes around each central node as an update of the feature representation of the current central node. When L rounds of iterations are complete, the final node representation is the output of the encoder. Such models can easily and effectively capture the local structure and affiliated attribute information of HINs. Currently, GNN-based models achieve state-of-the-art performance on many graph-based tasks, including natural language processing, knowledge graphs, and protein networks. According to the way information is propagated in a graph, GNNs are mainly divided into spectral-based graph convolution, spatial-based graph convolution, graph attention networks, and graph spatiotemporal networks, etc. The corresponding representative models for homogeneous networks are GCN [66], GraphSage [67], GAT [68], and GGNN [69], respectively.



(a) Collect neighborhood (b) Aggregate feature information from neighbors (c) Predict network structure and/or label using the aggregated features

Figure 8: Overview of the GNN models. The figures are adapted from [67]

For HINs, because it is usually necessary to process different types of semantic information, feature aggregation usually follows a hierarchical aggregation strategy. In general, HNE models first use the first-level aggregation function $AGG^N(\cdot)$ to aggregate the features of nodes within

each subnetwork of the same semantic relationship, followed by a second-level aggregation function $AGG^S(\cdot)$ to aggregate the node features for different semantic relations. The modeling ability of heterogeneous GNN networks differs depending on the underlying GNN methodology. In general, the spectral-based GCN models have a strong theoretical basis for graph signal processing, but because the features of all neighbor nodes are aggregated each time, the computational complexity of this type of model is high, and they are usually transductive models. The spatial-based GCN models can control the computational complexity of the model by sampling neighborhoods. Moreover, since their model input can be independent of the number of nodes in the network, this GNN model type is more flexible. The graph attention network models can easily solve the weight problem of neighbor nodes.

DMGI [70] is a spectral-based convolutional GNN model designed for multi-relational networks with heterogeneous attributes. The model first uses the GCN model [66] to aggregate the neighbor features of each node inside each subnetwork $\mathscr{G}^{(k)}$:

$$\boldsymbol{H}^{(k)} = \sigma\left(\hat{\boldsymbol{D}}_k^{-1/2}\hat{\boldsymbol{A}}^{(k)}\hat{\boldsymbol{D}}_k^{-1/2}\boldsymbol{X}\boldsymbol{W}^{(k)}\right), \tag{21}$$

where $\hat{\boldsymbol{A}}^{(k)} = \boldsymbol{A}^{(k)} + \omega\boldsymbol{I}_n, \hat{\boldsymbol{D}}_{ii} = \sum_j \hat{\boldsymbol{A}}_{ij}$. Then, it aggregates the embeddings of all nodes within each subgraph to generate a subgraph-level summary representation as: $\boldsymbol{s}^{(k)} = \sigma\left(\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{h}_i^{(k)}\right)$. Next, to capture the global structural information of the networks, the DMGI model uses the Deep Graph Infomax method [71] to define the loss function. the main idea of the DGI is to maximize the mutual information between the local patches of a graph and the global embedding of the subgraph, i.e.,

$$\mathscr{L}^{(k)} = \sum_{i=1}^{n}\left[\log\sigma\left(\boldsymbol{h}_i^{(k)^T}\boldsymbol{M}^{(k)}\boldsymbol{s}^{(k)}\right) + \sum_{j=1}^{n}\log\sigma\left(-\tilde{\boldsymbol{h}}_j^{(k)^T}\boldsymbol{M}^{(k)}\boldsymbol{s}^{(k)}\right)\right]. \tag{22}$$

The model then uses a consensus regularization framework to aggregate the relation-type specific node embeddings to generate the final consensus node embeddings. Owing to the introduction of the DGI method, the DMGI model can not only capture the attribute information and low-level structural information, but also effectively capture the global structural information of the HINs.

However, in many GNN-based models (such as DMGI), the neighbor nodes in the HINs are defined as immediate neighbors. Several studies have extended the definition of neighbor nodes in heterogeneous networks using meta-path-based neighbors [6,72–79]. Such models are able to capture more specific semantic information and overcome possible sparsity issues. The HAN model [6] shown in Fig. 9 is a typical one. The core idea of HAN is to use a metapath-augmented adjacency matrix to replace the original adjacency matrix. It first projects the initial node attribute matrix using the following type-specific transformation: $\boldsymbol{h}_i = \boldsymbol{M}_{\tau(v_i)}\boldsymbol{x}_i$, where $\boldsymbol{x}_i$ is the initial node attribute vector of node v_i, and $\boldsymbol{M}_{\tau(v_i)}$ is the type-specific projection matrix of node v_i. Then, based on the multiple meta-paths defined in the heterogeneous network, the model uses a two-level attention mechanism to aggregate the features of each node's neighbors. The first layer aggregates the neighbor node features of each node based on the meta-path ϕ:

$$\boldsymbol{z}_i^{\phi} = \sigma\left(\sum_{j\in\mathscr{N}^{\phi}(i)}\alpha_{ij}^{\phi}\boldsymbol{h}_j\right), \tag{23}$$

where $\alpha_{ij}^{\phi} = \frac{exp(e_{ij}^{\phi})}{\sum_{k\in\mathscr{N}^{\phi}(i)}exp(e_{ik}^{\phi})}$, and $e_{ij}^{\phi} = a_{\phi}^{T}\cdot\left[\boldsymbol{h}_i;\boldsymbol{h}_j\right]$. The second-level attention mechanism aggregates each node's feature representation based on multiple meta-paths, namely:

$$z_i = \sum_{\phi \in \Phi} \beta_\phi z_i^\phi, \tag{24}$$

where $\beta_\phi = softmax(\frac{1}{N}\sum_{v_i \in \mathcal{V}} \boldsymbol{q}^T \sigma(\boldsymbol{W} z_i^\phi + b))$ is the significance of the meta-path $\phi \in \Phi$.

The HAN model can well capture low-order and high-order structural information, as well as specific semantic information, and can distinguish different weights for different neighbor nodes and different semantic relations (different meta-paths). The GraphInception [72], MAGNN [76], RoHe [79], HAHE [73], Player2Vec [74], HDGI [77], and MEIRec [75] models share a similar idea with the HAN model. They all use adjacency matrices based on a specific meta-path to replace the adjacency matrix $\boldsymbol{A}$ in the traditional model and then use a multi-level GCN or GAT network to perform hierarchical aggregation of the neighbor features of nodes. The main differences between them are as follows: ROHE [79] uses a metapath-based probability transition matrix to calculate the confidence of node neighbors, thereby filtering unimportant neighbor nodes to improve the robustness of the model; GraphInception [72] uses the eigenvectors of the probability transition matrix $\boldsymbol{P}$ as the Fourier basis of the heterogeneous graph convolution to perform graph convolution operations; MAGNN [76] also considers intermediate nodes on meta-path instances when aggregating the features of metapath-based neighbors. Similar to DMGI, the HDGI model [77] also uses DGI to define the loss function, so it also has the ability to capture global structural information. MEIRec [75] uses CNN and LSTM networks as the aggregation function of node neighbors according to the different types of node neighbors.

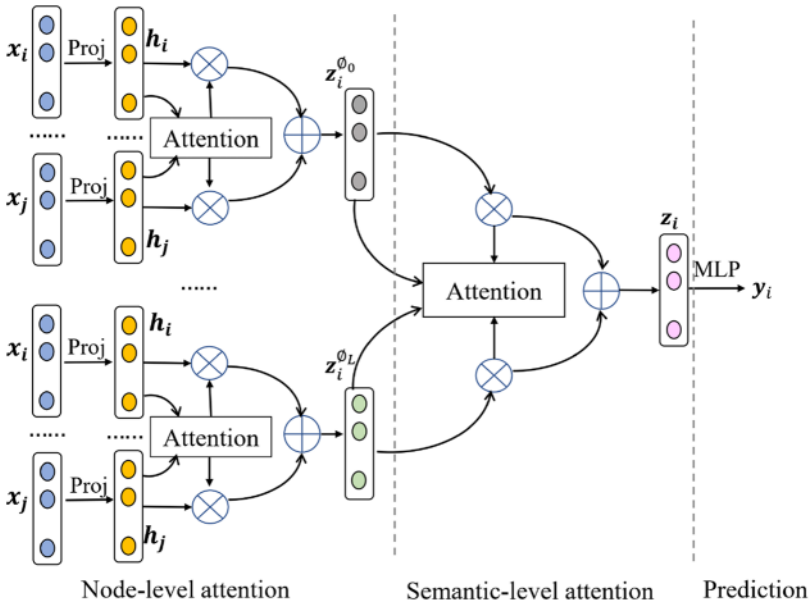


Figure 9: HAN model framework

Owing to rapid development, the current research on GNNs has shown explosive growth, and there have been many other GNN-related studies [78,80–90]. We list some typical GNN-based HNE models in Table 5. To be specific, the FAME model [87] improves the efficiency of heterogeneous spectral convolution models using sparse random projections; the HWNN model [78] uses wavelet basis instead of Fourier basis for graph convolution operation, avoiding the time-consuming Laplace matrix decomposition operation; when the HetGNN model [80] aggregates the information of neighbor nodes, it selects the top-k most important neighbor nodes based on the restart random walk strategy,

and then uses the two-level Bi-LSTM framework to aggregate the various types of content attribute features of neighbor nodes; the HGT model uses the multi-head attention mechanism to aggregate the information of neighbor nodes; the ActiveHNE [82] model is a semi-supervised spectral-based GCN model that introduces active learning techniques, which can effectively utilize the most valuable label information and reduce the workload of manual labeling by adding an active query module on the basis of the HNE module; different from other GNN-based models, the MEGNN model [90] mine meaningful meta-paths in heterogeneous networks through multilayer GNN information propagation operations.

Table 5: Graph neural network based models

Type	Model	$AGG^S(\cdot)$	Loss function	Preserve AT
Spectral-based GCN models	DMGI [70]	Attention	$\sum_{k=1}^{K}\mathscr{L}^{(k)}+\alpha\mathscr{L}_{cs}+\beta\mathscr{L}_{reg}$*	yes
	HDMI [86]	Attention	$\lambda_1 I(\boldsymbol{z}_i;\boldsymbol{s})+\lambda_2 I(\boldsymbol{z}_i;\boldsymbol{x}_i)+\lambda_3 I(\boldsymbol{z}_i;\boldsymbol{s};\boldsymbol{x}_i)$	yes
	HDGI [77]	Attention	$\frac{1}{N+M}\Big[\sum_{i=1}^{N}\mathbb{E}_{\boldsymbol{z}_i\sim\Delta^+}\log\sigma(\boldsymbol{z}_i^T\boldsymbol{W}_D\boldsymbol{s})+\sum_{j=1}^{M}\mathbb{E}_{\tilde{\boldsymbol{z}}_j\sim\Delta^-}\log\sigma(-\tilde{\boldsymbol{z}}_j^T\boldsymbol{W}_D\boldsymbol{s})\Big]$	yes
	FAME [87]	Weighted SUM	$\mathscr{L}_Y$	yes
	ActiveHNE [82]	CONCAT	$\mathscr{L}_Y$	yes
	Player2Vec [74]	Attention	$\mathscr{L}_Y$	yes
	HWNN [78]	CONCAT	$\mathscr{L}_Y+\lambda\mathscr{L}_{reg}$	yes
	Graph Incepetion [72]	CONCAT	$\mathscr{L}_Y$	yes
Spatial-based GCN models	GCMC [91]	CONCAT, SUM	$-\sum_{(u_i,v_j)\in\Delta^+}\sum_{r=1}^{R}I(r=\boldsymbol{M}_{ij})\log\frac{e^{\boldsymbol{z}_i{}^T\boldsymbol{Q}_r\boldsymbol{z}_j}}{\sum_{s\in R}\boldsymbol{z}_i{}^T\boldsymbol{Q}_s\boldsymbol{z}_j}$	yes
	MEIRec [75]	CONCAT	$\mathscr{L}_Y$	yes
	HetGNN [80]	Attention	$\sum_{v\in\mathscr{V}}\sum_{t\in\mathscr{T}}\sum_{v_c\in\mathscr{N}^t(v)}\log\sigma(z_v\cdot z_{v_c})+\mathbb{E}_{v_{c'}\sim P_n^t(v)}\log\sigma(-z_v\cdot z_{v_{c'}})$	yes
	HEP [81]	CONCAT	$\sum_{v\in\mathscr{V}}\mathbb{E}_{u\sim P_n(u)}\Big[\gamma+\\|\tilde{\boldsymbol{z}}_v-\boldsymbol{z}_v\\|_2^2+\\|\tilde{\boldsymbol{z}}_v-\boldsymbol{z}_u\\|_2^2\Big]_+\mathscr{L}_Y$	yes
Graph attention network models	HGAT [83]	/	$\mathscr{L}_Y$	yes
	HGT [84]	Multi-head GAT	$\mathscr{L}_Y$	yes
	DisenHAN [88]	GAT	$\mathscr{L}_{Rec}$	yes
	HAN [6]	Attention	$\mathscr{L}_Y$	yes
	MAGNN [76]	Attention	$-\sum_{(u,v)\in\Delta^+}\log\sigma(\boldsymbol{z}_u\cdot\boldsymbol{z}_v)-\sum_{(u',v')\in\Delta^-}\log\sigma(-\boldsymbol{z}_{u'}\cdot\boldsymbol{z}_{v'})+\mathscr{L}_Y$	yes
	RoHe [79]	Attention	$\mathscr{L}_Y$	yes
	HAHE [73]	Attention	$\mathscr{L}_Y$	no

Note: * $\mathscr{L}_{cs}=(\boldsymbol{Z}-\underset{r\in\mathscr{R}}{agg}\boldsymbol{H}^{(r)})^2-(\boldsymbol{Z}-\underset{r\in\mathscr{R}}{agg}\tilde{\boldsymbol{H}}^{(r)})^2$ is a consensus regularization loss.

In summary, GNN-based HNE models have a good ability to capture attribute information as well as low and high-order structural features. In addition, the input of spatial-based GCN models can be independent of the number of nodes in the current network, so such models are inductive models that can generate embeddings for nodes that are not currently observed. However, this type of model still face the following challenges:

- In a large-scale heterogeneous network, it is very resource intensive to aggregate the features of all the neighbor nodes of one node. Many GNN models use neighbor sampling to reduce model complexity. However, the distribution within different subnets in an HIN may vary substantially. Neighbor sampling must be carefully designed so that the sampling nodes include relatively important neighbor nodes, otherwise important network information may be lost.
- In the GNN model, the number of network layers is key to the performance of the model. A model that is too shallow cannot capture the high-order features of the network; however, blindly increasing the number of layers of a GNN network may degrade the performance of the model while increasing the training complexity of the model [92]. Because the graph convolution operation will make the feature representations of adjacent nodes increasingly more similar, in theory, when there are enough layers, the feature representations of all the nodes in a connected graph will converge to a single point [64], which is called the over-smoothing problem. In addition, when there are too many layers, the model will also amplify some noise due to the continuous iterative convolution operation, making the model more vulnerable to attack. Therefore, how to deal with the problem that the model cannot be deeper remains a challenge in GNN-based models.

3.5 Knowledge Graph Embedding-Based Models

In recent years, knowledge graph techniques have been rapidly developed. Numerous knowledge graphs, such as WordNet [93], Freebase [94], and Yago [95], have been successfully applied to many practical applications. A knowledge graph can be represented by the triple $\mathscr{G} = \{< h, r, t >\} \subseteq \mathscr{E} \times \mathscr{R} \times \mathscr{E}$, where h is the head entity, t is the tail entity, and $< h, r, t >$ indicates that the entities h and t are connected by a specific relationship $r \in \mathscr{R}$. KGE models aim to learn a function to map the entities and relations in the knowledge graph to a continuous vector space to capture the structural and semantic information contained in the knowledge graph [96]. As a knowledge graph can be viewed as an HIN, KGE models are types of HNE models. However, different from general HNE models, in order to deal with diverse semantic relations contained in the knowledge graph, KGE models not only encode the entity objects, but also encode specific relation types.

Most KGE models also use the direct encoding, and the decoder is always a ternary scoring function $f_r(h, t)$ that measures the acceptability of each triplet $< h, r, t >$. According to the definition of the scoring function, KGE models can be divided into two main categories: 1) translation distance-based models, which model the relationship as the distance transformation from the head to tail entity, and the transformed distance difference defines the scoring functions, and 2) semantic matching models, which use similarity-based scoring functions to measure the credibility of facts by matching the latent semantics of entities and relationships contained in the vector space representation.

3.5.1 Translation Distance-Based KGE Models

TransE [97] is a typical translation distance-based KGE model. Its basic idea is to regard the relationships in the knowledge graph as a transformation from the head to tail entity. Specifically, it assumes that if the triple $< h, r, t >$ holds, then the embedding of the tail entity t should be close to the embedding of the head entity h plus some vector that depends on the relationship r, i.e., $\boldsymbol{h} + \boldsymbol{r} \approx \boldsymbol{t}$. The scoring function is therefore defined as

$$f_r(h, t) = -\|\boldsymbol{h} + \boldsymbol{r} - \boldsymbol{t}\|_{1/2}. \tag{25}$$

The loss function of the TransE model is a margin-based ranking loss defined on the training set:

$$\mathscr{L} = \sum_{(h,r,t)\in\Delta} \sum_{(h',r,t')\in\Delta'} [\gamma + f_r(h,t) - f_r(h',t')]_+. \tag{26}$$

TransE is very simple and efficient but does not handle one-to-many and many-to-many relationships well. To overcome the shortcomings of TransE, models such as TransH [98], TansR [99], TransD [100] extend the TransE model by allowing entities to have different embeddings for different relations. The simple illustrations of TransE, TransH and transR models are shown in Fig. 10. Furthermore, KG2E [101], TransG [102], and SE [103] are also translation distance-based KGE models. Unlike other models, KG2E and TransG assume that relations and semantics are inherently uncertain, and they use Gaussian distributions to model the entities and relations in knowledge graphs.

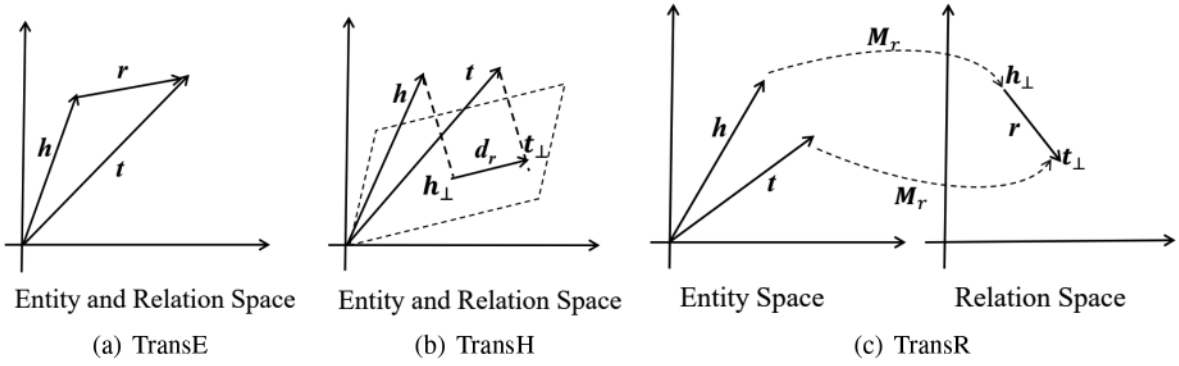


Figure 10: Simple illustrations of TransE, TransH, and TransR models. The figures are adapted from [96]

3.5.2 Semantic Matching KGE Models

The RESCAL model [104] is a typical semantic matching model. In this model, each entity is represented as a vector and each relation r is represented as a relation matrix $\boldsymbol{M}_r$, which models the pairwise interactions between latent vectors. The scoring function of the model is a bilinear function as

$$f_r(h,t) = \boldsymbol{h}^T M_r \boldsymbol{t}, \tag{27}$$

The DistMult model [105] simplifies the matrix $\boldsymbol{M}_r$ and restricts it to a diagonal matrix, i.e., $\boldsymbol{M}_r = diag(\boldsymbol{r})$. However, due to $\boldsymbol{h}^T diag(\boldsymbol{r})\boldsymbol{t} = \boldsymbol{t}^T diag(\boldsymbol{r})\boldsymbol{h}$, the Distmult model can only deal with symmetric relations.

In addition to RESCAL and its extended models, some semantic matching models employ neural network-based frameworks to semantically match entities and relationships [106,107]. ConvE [106] is a semantic matching KGE model that uses a multilayer CNN architecture. The model mainly consists of an encoder and a scorer. For the input triplet $< h, r, t >$, the encoder projects the head entity h and the tail entity t into d-dimensional space, resulting in $\boldsymbol{h}$ and $\boldsymbol{t}$. The scorer first converts the vectors of head entity embedding $\boldsymbol{h} \in \mathbb{R}^d$ and relation embedding $\boldsymbol{r} \in \mathbb{R}^d$ into second-order vectors: $\bar{\boldsymbol{h}}, \bar{\boldsymbol{r}} \in \mathbb{R}^{d_w \times d_h}$, where $d = d_w \times d_h$. Then, $\bar{\boldsymbol{h}}$ and $\bar{\boldsymbol{r}}$ are concatenated and input to a convolutional layer for feature extraction. Next, the model flattens and linearly transforms the extracted feature tensors. The framework of the ConvE model is shown in Fig. 11. The scoring function is defined as follows:

$$f_r(h,t) = \sigma(vec(f([\bar{\boldsymbol{h}};\bar{\boldsymbol{r}}]) * \omega)\boldsymbol{W})\boldsymbol{t}. \tag{28}$$

Due to the introduction of CNN in the KGE model, the expressive power of the ConvE model is enhanced, and the number of model parameters is controlled.

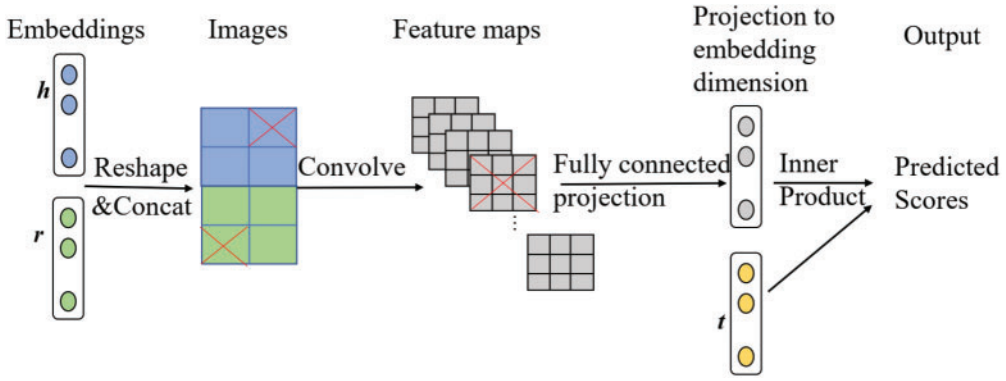


Figure 11: The ConvE model framework. In this model, the entity and relation embeddings are first reshaped and concatenated (Step 1); and then the resulting matrix is then used as input to a convolutional layer (Step 2); the resulting feature map tensor is vectorised and projected into a d-dimensional space (Step 3), and matched with the candidate object embeddings (Step 4). The figure is adapted from [106], and the symbol " $\times$ " represents dropout

NTN [107] is another semantic matching model. It uses a neural network structure to describe each triple and gives a score, which is an extension of the SLM model [107]. For a given triple $< h, r, t >$, the scoring function for NTN is defined as

$$f_r(h,t) = \boldsymbol{r}^T \sigma(\boldsymbol{h}^T \hat{\boldsymbol{W}}_r \boldsymbol{t} + \boldsymbol{M}_r \begin{pmatrix} \boldsymbol{h} \\ \boldsymbol{t} \end{pmatrix} + \boldsymbol{b}_r), \tag{29}$$

where $\hat{\boldsymbol{W}}_r \in \mathbb{R}^{d \times d \times k}$ is a relation-r-specific tensor, and $\boldsymbol{M}_r \in \mathbb{R}^{k \times 2d}$ is the relation-r-specific weight matrix. The NTN model is very expressive. However, this model contains a large number of model parameters for each relation, thus affecting the efficiency of the model.

We summarize the basic characteristics of some of the most common KGE models in Table 6. For more other KGE models, we refer the reader to the KGE review [96]. In general, the main difference between KGE and other types of models is that KGE models are able to generate representations for relations in an explicit manner, thus more fully expressing the heterogeneity of relational semantics in the HINs. However, the basic KGE model has the following defects:

- The encoder usually has the disadvantage of direct encoding.
- Such models usually only consider the low-order neighbor relationships when modeling the structural information of the HINs.
- Further, such models usually do not model attribute information.

In order to improve the quality of generated embeddings, some KGE models extend the basic KGE model [110–112], and some models combine more other network information when generating embeddings, including entity attribute information [113], relations path [114], logic rules, and supervision information for some downstream ML tasks. Common downstream ML tasks related to knowledge graphs mainly include knowledge graph completion, link prediction, and recommender systems [115], etc.

Table 6: Knowledge graph embedding models

Model	Entity embeddings	Relation embedding	Score fucntion	Limitations
TransE [97]	$\boldsymbol{h}, \boldsymbol{t} \in \mathbb{R}^d$	$\boldsymbol{r} \in \mathbb{R}^d$	$\|\boldsymbol{h}+\boldsymbol{r}-\boldsymbol{t}\|_{l1/2}$	$\|\boldsymbol{h}\|_2^2 = 1, \|\boldsymbol{t}\|_2^2 = 1$
TransH [98]	$\boldsymbol{h}, \boldsymbol{t} \in \mathbb{R}^d$	$\boldsymbol{r}, \boldsymbol{w}_r \in \mathbb{R}^d$	$\left\|(\boldsymbol{h}-\boldsymbol{w}_r^T\boldsymbol{h}\boldsymbol{w}_r)+\boldsymbol{d}_r -(\boldsymbol{t}-\boldsymbol{w}_r^T\boldsymbol{t}\boldsymbol{w}_r)\right\|_2^2$	$\|\boldsymbol{w}_r\|_2 = 1$
TransR [99]	$\boldsymbol{h}, \boldsymbol{t} \in \mathbb{R}^d$	$\boldsymbol{r} \in \mathbb{R}^k, \boldsymbol{M}_r \in \mathbb{R}^{k\times d}$	$\|\boldsymbol{h}\boldsymbol{M}_r+\boldsymbol{r}-\boldsymbol{t}\boldsymbol{M}_r\|_{1/2}$	$\|\boldsymbol{h}\|_2^2 \le 1, \|\boldsymbol{t}\|_2^2 \le 1, \|\boldsymbol{r}\|_2^2 \le 1, \|\boldsymbol{h}\boldsymbol{M}_r\|_2^2 \le 1, \|\boldsymbol{t}\boldsymbol{M}_r\|_2^2 \le 1$
TransD [100]	$\boldsymbol{h}, \boldsymbol{t}, \boldsymbol{h}_p, \boldsymbol{t}_p \in \mathbb{R}^d$	$\boldsymbol{r}, \boldsymbol{r}_p \in \mathbb{R}^d$	$\left\|\boldsymbol{h}\boldsymbol{M}_{rh}+\boldsymbol{d}_r-\boldsymbol{t}\boldsymbol{M}_{rt}\right\|_{1/2}$	$\|\boldsymbol{h}\|_2^2 \le 1, \|\boldsymbol{t}\|_2^2 \le 1, \|\boldsymbol{r}\|_2^2 \le 1, \left\|\boldsymbol{h}\boldsymbol{M}_{rh}\right\|_2^2 \le 1, \|\boldsymbol{t}\boldsymbol{M}_{rt}\|_2^2 \le 1$
KG2E [101]	$\boldsymbol{h}, \boldsymbol{t} \in \mathbb{R}^d$	$\boldsymbol{r} \in \mathbb{R}^d$	$\frac{1}{2}(\mu^T\Sigma^{-1}\mu+\log det\Sigma)$, $\Sigma = \Sigma_h+\Sigma_t+\Sigma_r$	$\|\boldsymbol{h}\|_2 \le 1, \|\boldsymbol{r}\|_2 \le 1, c_{min}I \le \Sigma_h, \Sigma_t, \Sigma_r \le c_{max}I, c_{min}I > 0$
RESCAL [104]	$\boldsymbol{h}, \boldsymbol{t} \in \mathbb{R}^d$	$\boldsymbol{M}_r \in \mathbb{R}^{d\times d}$	$\boldsymbol{h}^T\boldsymbol{M}_r\boldsymbol{t}$	$\|\boldsymbol{h}\|_2 \le 1, \|\boldsymbol{t}\|_2 \le 1, \|\boldsymbol{M}_r\|_F \le 1$
DistMult [105]	$\boldsymbol{h}, \boldsymbol{t} \in \mathbb{R}^d$	$\boldsymbol{r} \in \mathbb{R}^d$	$\boldsymbol{h}^T diag(\boldsymbol{r})\boldsymbol{t}$	$\|\boldsymbol{h}\|_2 \le 1, \|\boldsymbol{t}\|_2 \le 1, \|\boldsymbol{r}\|_2 \le 1$
ConvE [106]	$\boldsymbol{h}, \boldsymbol{t} \in \mathbb{R}^d$	$\boldsymbol{r} \in \mathbb{R}^d$	$\sigma(vec(f([\bar{\boldsymbol{h}};\bar{\boldsymbol{r}}])*\omega)\boldsymbol{W})\boldsymbol{t}$	$\bar{\boldsymbol{h}}, \bar{\boldsymbol{r}} \in \mathbb{R}^{k_w\times k_h}, k = k_w \times k_h$
SLM [107]	$\boldsymbol{h}, \boldsymbol{t} \in \mathbb{R}^d$	$\boldsymbol{\mu}_r \in \mathbb{R}^d, \boldsymbol{M}_{r1}, \boldsymbol{M}_{r2} \in \mathbb{R}^{k\times d}$	$\boldsymbol{\mu}_r^T f(\boldsymbol{M}_{r1}h+\boldsymbol{M}_{r2}t+\boldsymbol{b}_r)$	$\|\boldsymbol{h}\|_2 \le 1, \left\|\boldsymbol{\mu}_r\right\|_2 \le 1, \|\boldsymbol{t}\|_2 \le 1, \|\boldsymbol{M}_{r1}\|_F \le 1, \|\boldsymbol{M}_{r2}\|_F \le 1$
NTN [107]	$\boldsymbol{h}, \boldsymbol{t} \in \mathbb{R}^d$	$\boldsymbol{\mu}_r, \boldsymbol{b}_r \in \mathbb{R}^k, \boldsymbol{M}_{r1}, \boldsymbol{M}_{r2} \in \mathbb{R}^{k\times d}, \hat{\boldsymbol{M}}_r \in \mathbb{R}^{d\times d\times k}$	$\boldsymbol{\mu}_r^T f(\boldsymbol{h}^T\hat{\boldsymbol{M}}_r t+\boldsymbol{M}_{r1}\boldsymbol{h}+\boldsymbol{M}_{r2}\boldsymbol{t}+\boldsymbol{b}_r)$	$\|\boldsymbol{h}\|_2 \le 1, \left\|\boldsymbol{\mu}_r\right\|_2 \le 1, \|\boldsymbol{t}\|_2 \le 1, \|\boldsymbol{M}_{r1}\|_F \le 1, \|\boldsymbol{M}_{r2}\|_F \le 1, \|\boldsymbol{b}_r\|_2 \le 1$
RotatE [108]	$\boldsymbol{h}, \boldsymbol{t} \in \mathbb{C}^d$	$\boldsymbol{r} \in \mathbb{C}^d$	$\|\boldsymbol{h}\odot\boldsymbol{r}-\boldsymbol{t}\|_{1/2}$	$\|\boldsymbol{r}_i\| = 1$
HolE [109]	$\boldsymbol{h}, \boldsymbol{t} \in \mathbb{R}^d$	$\boldsymbol{r} \in \mathbb{R}^d$	$\boldsymbol{r}^T(\boldsymbol{h}*\boldsymbol{t})$	$\|\boldsymbol{h}\|_2 \le 1, \|\boldsymbol{r}\|_2 \le 1, \|\boldsymbol{t}\|_2 \le 1$

3.6 Hybrid Models

To leverage the advantages of multiple techniques, some hybrid models have been proposed.

3.6.1 GNN + RW

Some models fuse graph nueral networks and random walk techniques to generate network embeddings. The typical idea of this type of model is to use GNN as the encoder instead of the direct encoding function in the common random walk-based models [116,117].

GATNE [116] is one such type of hybrid model. The idea of GATNE is similar to the MNE model [41], but it is more general. Specifically, GATNE first uses the GNN framework to aggregate the neighbor node features of each node within each sub-network $\mathscr{G}^r$, and obtaines the aggregated features of each node v_i, namely $\boldsymbol{u}_i^r = \underset{j\in\mathscr{N}^r(i)}{agg}(\boldsymbol{u}_j^r)$. Next, based on the obtained aggregated features, the model represents the embedding of each node v_i in each sub-network $\mathscr{G}^r$ as the sum of the base embedding $\boldsymbol{b}_i$ and the relation r-specific node embeddings, i.e.,

$$\boldsymbol{z}_i^r = \boldsymbol{b}_i + \alpha_r \cdot \boldsymbol{M}_r^T \cdot \boldsymbol{U}_i \cdot \boldsymbol{a}_i^r, \tag{30}$$

where $\boldsymbol{U}_i = stack(\boldsymbol{u}_i^1, ...\boldsymbol{u}_i^r, ..., \boldsymbol{u}_i^{|\mathscr{R}|}) \in \mathbb{R}^{d_0*|\mathscr{R}|}$, α_r is a hyperparameter denoting the importance of the edge embeddings toward the overall embedding, and $\boldsymbol{M}_r$ is a trainable weighting matrix. The model then samples a large number of random walk sequences based on a specific meta-path defined in the HIN, and uses the skip-gram algorithm to solve the optimized node embeddings. It is worth mentioning that,

besides the transduction version GATNE-T, the authors also proposed an inductive version GATNE-I which can generate embeddings for nodes that are currently unobserved in the HINs.

Due to the introduction of the GNN encoder in the random walk-based model, the GATNE model can not only flexibly encode low-order and higher-order structural information, but also capture the attribute information of the network well.

3.6.2 GNN + KGE

As mentioned in Section 3.5, traditional KGE models are usually not good at handling higher-order structural information and attribute information. Some studies used the GNN encoder to replace the direct encoding function in the basic KGE model to improve the modeling ability of the HNE models [118–121].

LinkNBed is a KGE model that incorporates the idea of GNN. The model contains three layers: atomic, context, and final representation. At the atomic level, the model directly encodes all entities, relationships, entity types, and attribute information contained in the knowledge graph as follows:

$$\begin{aligned} &\boldsymbol{h} = \sigma(\boldsymbol{W}^E \boldsymbol{v}_h), \boldsymbol{r} = \sigma(\boldsymbol{W}^R \boldsymbol{v}_r) \\ &\boldsymbol{t} = \sigma(\boldsymbol{W}^E \boldsymbol{v}_t), \boldsymbol{T} = \sigma(\boldsymbol{W}^l \boldsymbol{v}_l), \end{aligned} \tag{31}$$

where $\boldsymbol{W}^E$, $\boldsymbol{W}^R$, and $\boldsymbol{W}^l$ are the projection matrices of entities, relations, and node types, respectively. At the context layer, the model uses an attention mechanism to aggregate contextual feature information of various objects as follows:

$$N_c(z) = \underset{\forall z' \in \mathcal{N}(z)}{agg} (q(z') \times \boldsymbol{z}'), \tag{32}$$

where z can be an entity or relationship and $q(z')$ is the weight calculated using the attention mechanism. In the final representation layer, the model aggregates the object's own feature representation, context feature representation, and attribute feature representation to obtain the final representation of the entities and relationships. The scoring function for each tuple $< h, r, t >$ is defined as

$$f_r(\boldsymbol{z}_h, \boldsymbol{z}_t) = \sigma\left(\sum_{q=1}^{d} \boldsymbol{z}_h \odot \boldsymbol{z}_r \odot \boldsymbol{z}_t\right). \tag{33}$$

Because of the introduction of GNN encoders based on KGE models, this model can effectively capture more information in HINs, including low-order, high-order structural information, relational semantic information, and attribute information.

Similar to the LinkNBed model, the NKGE [120], CACL [121], SACN [119], LightCAKE [122] and HRAN [123] models are also KG+GNN-type hybrid models. The main difference among these models is the type of encoder or decoder: LinkNBed and LightCAKE use a GAT network when aggregating neighbor node information, SACN uses a GCN model, NKGE uses a deep memory network, and HARN uses a two-level GNN. When defining the ternary scoring function, LinkNBed uses a simplified version of the bilinear function, LightCAKE uses TransE/DistMult, SACN and NKGE use ConvE models, whereas CACL and HRAN use CNN-based deformation models.

3.6.3 GNN + AE

To reduce the influence of noise and enhance the robustness of the model, some studies fuse the VAE framework with GNNs [124,125].

The RELEARN model [124] shown in Fig. 12 is a VAE framework designed for relational learning in a multi-relational heterogeneous social network. This model firstly represents the edge embedding $\boldsymbol{h}_{ij}$ for each edge e_{ij} as a weighted sum of K independent global relation embeddings $\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_K$ (each $\boldsymbol{w}_k$ follows a Gaussian distribution), i.e.,

$$\boldsymbol{h}_{ij} = \sum_{k=1}^{K} \boldsymbol{z}_{ijk} \boldsymbol{w}_{ijk}, \tag{34}$$

where $\boldsymbol{z}_{ijk}$ is the relation factor obeying the K multinomial distribution, and it can be obtained by the following steps: firstly, the model use the GCN framework to aggregate the neighbor features of each node to obtain the aggregated node embeddings; next, for each edge $e_{ij} = (v_i, v_j)$, the model concatenates the aggregated feature representations of the two nodes v_i and v_j, and then inputs the concatenated vector $\boldsymbol{y}_{ij} = [\boldsymbol{u}_i; \boldsymbol{u}_j]$ to a feedforward neural network to obtain the relation factor $\boldsymbol{z}_{ij} = f_r(\boldsymbol{y}_{ij})$. In the decoding stage, the model uses the VAE framework to attempt to reconstruct various types of information of the network from the learned edge embeddings. The decoder network of RELEARN consists of three decoders, including a structural information decoder (E), an attribute information decoder (A), and an information diffusion decoder (D).

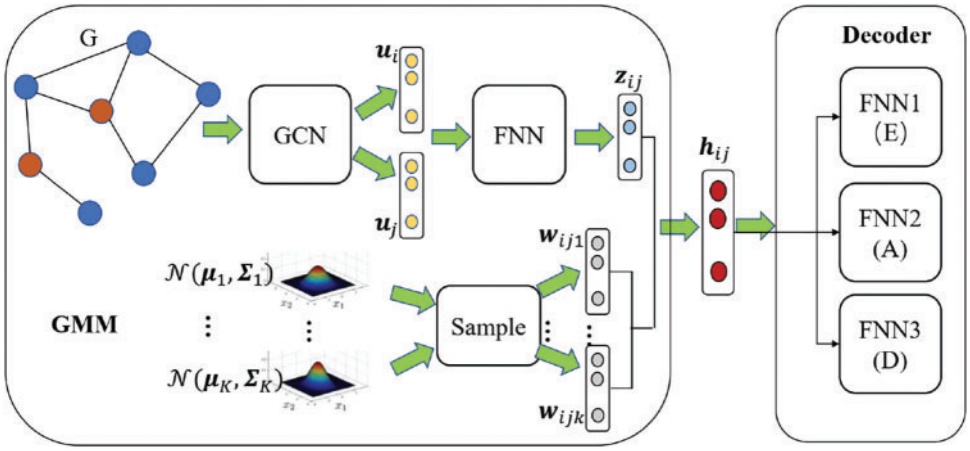


Figure 12: RELEARN model framework

Due to the use of the VAE framework, the RELEARN model can not only capture various types of information in the network, but also has good robustness.

3.6.4 Other Hybrid Models

Besides the above categories of hybrid models, some other types of hybrid models have been proposed, such as the KGE + RW models [126]. Moreover, some emerging hybrid models use reinforcement learning [127] or adversarial learning [128,129] frameworks. We provide a summary of some common hybrid HNE models in Table 7.

Table 7: Hybrid models

Type	Model	Encoder	Decoder	Loss function	Preserve ST	Preserve AT
GNN + RW	GATNE [116]	Generalized GCN	$softmax(\boldsymbol{z}_i^{(r)} \cdot \boldsymbol{c}_j)$	$-\sum_{v_i \in \mathcal{V}} \sum_{t \in \mathcal{T}} \sum_{v_j \in N_t(v_i)} \hat{s}_{ij} \log DEC(\boldsymbol{z}_i, \boldsymbol{z}_j)$	$1^{st} \sim n^{th}$	yes
	HGCN [117]	GCN	$softmax(\boldsymbol{z}_i \cdot \boldsymbol{c}_j)$	$-\sum_{v_i \in \mathcal{V}} \sum_{t \in \mathcal{T}} \sum_{v_j \in N_t(v_i)} \hat{s}_{ij} \log DEC(\boldsymbol{z}_i, \boldsymbol{z}_j)$	$1^{st} \sim n^{th}$	yes
GNN + KGE	LinkNBed [118]	GNN	$\sigma(\sum_{q=1}^{d} \boldsymbol{z}_h \odot \boldsymbol{z}_r \odot \boldsymbol{z}_t)$	$\alpha \mathcal{L}_{s1} + \beta \mathcal{L}_Y + \lambda \mathcal{L}_{Reg}$ [a]	$1^{st} \sim n^{th}$	yes
	SACN [119]	WGCN	Conv-TransE	$\mathcal{L}_{s2}$ [b]	$1^{st} \sim n^{th}$	yes
	NKGE [120]	Deep Memory Network	TransE/ConvE	$\mathcal{L}_{s1}$ or $\mathcal{L}_{s2}$	$1^{st} \sim n^{th}$	yes
	CACL [121]	CNN	$\sigma(\boldsymbol{p}^T \cdot (\boldsymbol{e}_c \otimes \boldsymbol{r}))$	$\mathcal{L}_Y$	$1^{st} \sim n^{th}$	no
	Light-CAKE [122]	GNN	TransE/DistMult	$\mathcal{L}_{s2}$	$1^{st} \sim n^{th}$	no
	HRAN [123]	GCN	ConvD	$\mathcal{L}_{s2}$	$1^{st} \sim n^{th}$	yes
GNN + AE	RELEARN [124]	GCN	FNN	$\sum_{X \in \{E,A,D\}} \mathbb{E}_{q_\phi}(\boldsymbol{Z}, \boldsymbol{W}, \boldsymbol{H}\|X) \cdot \log p_\theta(X\|H) - KL_{part}$	$1^{st} \sim n^{th}$	yes
	HeteHG-VAE [125]	GCN	$\sigma(\boldsymbol{z}_i^{\mathcal{V}^k} \cdot \boldsymbol{z}_j^{\mathcal{E}})$	$\sum_{k=1}^{K} \mathbb{E}_{q_\phi}[\log p_\theta(\boldsymbol{H}^{\mathcal{V}^k} \| \boldsymbol{Z}^{\mathcal{V}^K}, \boldsymbol{Z}^{\mathcal{E}}; \lambda^{\mathcal{V}^k})] - KL_{part}$	$1^{st} \sim n^{th}$	yes
RW + KGE	PRE [126]	DIRECT	$softmax(\boldsymbol{z}_i \cdot \boldsymbol{z}_j)$, KGE	$\sum_{v_i \in \mathcal{V}} \sum_{v_j \in \mathcal{N}(v_i)} [-\log p(v_j\|v_i) + \mathcal{L}_{KGE}(v_i, v_j)]$	$1^{st} \sim n^{th}$	no

Note: [a] $\mathcal{L}_{s1} = \sum_{(h,r,t) \in \Delta^+} \sum_{(h',r,t') \in \Delta^-} \left[\xi + f_r(\boldsymbol{h}, \boldsymbol{t}) - f_r(\boldsymbol{h}', \boldsymbol{t}')\right]_+$.
[b] $\mathcal{L}_{s2} = -\frac{1}{N} \sum_i y_i \log \sigma(f_r(\boldsymbol{h}, \boldsymbol{t})) + (1 - y_i) \log(1 - \sigma(f_r(\boldsymbol{h}, \boldsymbol{t})))$.

3.7 Summary

Sections 3.1 to 3.6 present an overview of the six existing HNE model types. For each type, we list the main characteristics of some representative models and use detailed tables to analyze and compare the basic elements of each model. It can be seen that the existing six models have obvious differences in encoder, decoder and empirical proximity matrix. At the same time, we also analyze their modeling ability in terms of capturing structural, relational semantic and attribute network information.

We summarize the advantages and disadvantages of these models in Table 8. From the perspective of modeling capability, MF models and KGE models are good at modeling low-order structural features, whereas other types are good at modeling both low- and high-order structural features. Furthermore, the KGE model has stronger semantic modeling capabilities, whereas GNN models have good ability to capture the attributes and higher-order structural information of the network. Hybrid models are usually able to model more network information by leveraging the advantages of multiple models.

Table 8: Summary of advantages and disadvantages of various types of models

Model type	Advantages	Disadvantages
MF	Shallow model, easy to implement, higher operating efficiency	(a) For large-scale networks, the decomposition of large dense higher-order adjacency matrices is intractable. (b) Attribute information usually needs to be handled separately. (c) Not good at handling different semantic information
RW	Edge sampling can reduce the model complexity; flexible; convenient to model low-order and high-order network structures	(a) It is prone to skewness problems and requires careful design of sampling strategies. (b) Defining meta-paths needs human prior experience. (c) Attribute information usually needs to be handled separately
AE	Many off-the-shelf AE frameworks are available; capable of capturing 1–2 order structural information	(a) Too many parameters in the multilayer AE framework and the training cost is high; (b) Not good at dealing with complex and diverse semantic information; (c) Attribute information usually needs to be handled separately
GNN	Deep learning-based model; can easily and effectively capture the attribute, low- or high-order structure information in the networks	(a) Over-smoothing is prone to occur as the number of convolutional layers increases, and the model will be more easily disturbed and attacked. (b) Inappropriate neighbor sampling functions will lead to loss of important network information
KGE	Can model many different types of semantic relations well; many ready-made KGE models that can be easily integrated with other technologies	(a) Traditional KGE models only model low-order proximities; (b) Attribute information usually needs to be handled separately
HB	In general, hybrid models are able to model more information in the network	Complexity of the hybrid model and the training cost typically increases compared to other types

4 Application Fields, Benchmark Datasets, Open Source Code, and Performance Comparison

This section focuses on application fields, benchmark datasets, open source tools, and performance evaluations for HNE.

4.1 Application Fields and Evaluation Metrics

Once new vertex representations are learned via HNE techniques, they can be used to solve various subsequent machine learning tasks. HNE benefits various graph analytics applications such

as node classification, node clustering, link prediction, recommender systems, visualization (Fig. 13). Meanwhile, the effectiveness of representation learning can also be validated by evaluating their performance on these various subsequent tasks [12].

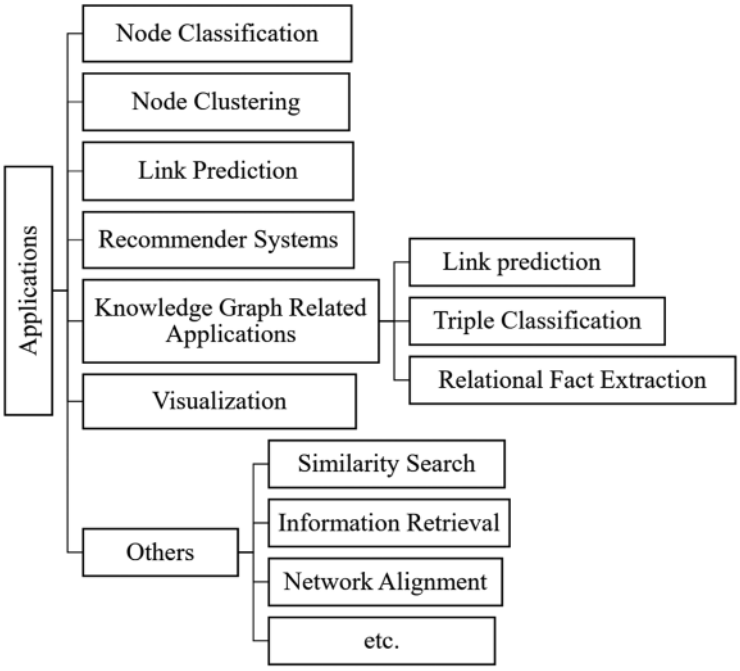


Figure 13: Application fields of HNE

4.1.1 Node Classification

Node classification is perhaps the most common benchmark task for evaluating node embeddings. Typically, network vertices are partially or sparsely labeled owing to high labeling costs, and most vertices in a network have unknown labels. The vertex classification problem aims to predict the labels of unlabeled vertices given a partially labeled network, e.g., determining the research field or affiliation information of an author. A good vertex representation improves the accuracy of node classification.

According to the classification task, node classification can be further divided into binary-class classification, multi-class classification, and multi-label classification (each node is associated with one or more labels from a finite label set). In the application of HNE, most cases are multi-class classification [4,21,45,52,72,112] or multi-label classification [5,21,39,48,49,52,61,62,72,80], and these models tipically use the Micro_F1 or Macro_F1 as the performance evaluation metric [5,6,21,26,30,32,38,48,49,52,59,61,70,73,76,77,80,86,87,112,127,129]. A few models use the average accuracy (ACC) [41,45] or the mean average precision (MAP) [62] as the evaluation metric.

4.1.2 Node Clustering

Node clustering is also a very common application of HNE models. Its goal is to cluster similar nodes into the same cluster and dissimilar nodes into different clusters as much as possible. Node clustering has a wide range of uses in bioinformatics, computer science, and sociology; for example, clustering a group of proteins with the same function in a biological network or clustering groups of people with similar interests in a social network into a cluster.

For simplicity, most HNE studies choose k-means as the clustering algorithm [4,6,32,36,38,39,45,52,61,76,80,86,112]. Node clustering is an unsupervised approach and does not use label information during model training; therefore, when evaluating the performance of node clustering tasks, for most models, the normalized mutual information [4–6,35,36,38,39,45,52,61,70,76,77,80,86,112,129] and the adjusted Rand index [5,6,36,39,52,76,77,80] are selected as the performance evaluation metrics.

4.1.3 Link Prediction

Link prediction is a very important ML task in graph mining systems. Its goal is to predict possible connections or existing but not observed relationships in current heterogeneous networks. Link prediction techniques can discover implicit or missing interactions in networks, identify false links, and understand network evolution mechanisms. For example, predicting unknown connections between people in social networks can be used to recommend friendships or identify suspicious relationships. Alternatively, link prediction methods are used in biological networks to predict previously unknown interactions between proteins, thereby significantly reducing the cost of empirical methods. A good network representation should be able to capture both explicit and implicit connections between network vertices, enabling applications to predict links.

Link prediction is often viewed as a simple binary classification problem: for any two potentially linked objects, the task is to predict whether a link exists (1) or does not exist (0). The vast majority of existing link prediction tasks use the area under the receiver operating characteristic curve (AUC) as the performance evaluation metric [30,31,36,39,41,42,48,52,59,60,62,76,80,110,112,116,130–132]. In addition, some studies use the F1-score [58,60,80,112,116], average precision [36,49,53,58,60,76,132], or recall [50,60] as the performance evaluation metric for link prediction tasks.

4.1.4 Recommender Systems

HNE is also frequently used in Recommender Systems (RSs). RSs can significantly enhance the commercial value of enterprises and effectively reduce the information overload of users. For over a decade, many businesses and companies have used RSs to recommend friends, products, and services to their customers [27,43,47,88,91,115]. Recommendation tasks are closely related to link prediction, but the main difference is that link prediction outputs a binary prediction, whereas RSs output a ranking list of length K.

For RSs, the most commonly used performance evaluation metrics are precision@k [27,34,47,58,88,115], recall@k [27,34,40,47,50,58,88,115], MAP@k [34,40,47], and NDCG@k [34,47,88,115,129]. A few models use other metrics, such as *AUC* [75,130], Hit Ratio@k [115,129,133], and RMSE score [46,91].

4.1.5 Knowledge Graph Related Applications

Compared to general HINs, a knowledge graph has more abundant heterogeneous entity types and semantic relation types. The main application tasks related to KGE are link prediction, triple classification, and relational fact extraction.

Link prediction in a knowledge graph is a typical task of predicting an entity that has a specific relation with another given entity, i.e., predicting h given (r, t) or t given (h, r), where the former is denoted as $(?, r, t)$ and the latter is denoted as $(h, r, ?)$. Essentially, link prediction in knowledge graphs is a knowledge graph completion task, i.e., the task of adding missing knowledge to the graph. The most commonly used indicators for knowledge graph link prediction tasks are mean rank [97–99,101,102,106,108,114,115,120,121], MRR [105,106,108,113,118–121,134], and Hits@n

[97–102,105,106,108,113–115,118–121,132,134]. Triple classification consists of verifying whether an unseen triple fact (h, r, t) is true or not, e.g., (Pablo Picaso, nationality, Spain) should be classified as a true fact, whereas (Pablo Picaso, nationality, United States) is a false one [107]. Accuracy is generally used as the evaluation metric in the triple classification task [98–102,107,121]. Relation extraction aims to extract relational facts from plain text in which entities have been detected. For example, given the phrase "Paris is in France," where the entities "h = Paris" and "t = France" have been detected, the relation extractor should predict that the relation between these two entities is "is in." For the relation extraction task, the precision–recall curve [98,99,114] is mainly used as the performance evaluation metric.

4.1.6 Visualization

The problem of graph visualization on 2D interfaces has been studied for a long time, with applications in areas such as biology, sociology, and data mining. Researchers can easily leverage existing general-purpose techniques to visualize high-dimensional datasets, which are useful for mining communities and other hidden structures. Common graph visualization approaches include applying a dimensionality reduction technique such as principal component analysis (PCA) [124] or t-distributed stochastic neighbor embedding (t-SNE) [4–6,26,31,51,61,72,73,76,86,88,125,129] to plot the node embedding vectors generated by the HNE models in a 2D space with different colors indicating the nodes' categories.

4.1.7 Other Applications

Besides the most common HNE applications discussed above, the applications of HNE models also include similarity search [45,51,70,86], information retrieval [50,130], network alignment [29,81], user profiling [83], and anomaly detection [63].

4.2 Benchmark Datasets

Benchmark datasets play a critical role in the evaluation of HNE models. According to the types of heterogeneous networks contained in the datasets, the commonly used benchmark datasets in existing HNE models can be divided into categories such as social networks, bibliographic networks, text networks, biological networks, knowledge graphs.

DBLP[2] and Aminer[3] [135] are the most frequently used bibliographic network datasets [4,6,26,28,30,32,36,38,48,49,51,52,59,61,70,72,73,76,78,79,82,86,112], whereas some other models use ACM[4] [6,32,48,70,72,77,79,86], Cora [78,82,104,126], DBIS [4,39], or other datasets. Moreover, the most frequently used social network datasets are Twitter[5] [26,28,30,41,60,128], Flickr[6] [5,30,62,130,136], and Gowalla[7] [27]. Among the commercial networks, the Yelp[8] and Amazon[9] datasets have been widely used [36,39,46–49,51,61,73,88,110]. 20NG[10] and WikiPedia[11] are commonly

[2] http://dblp.uni-trier.de/xml/.
[3] https://www.aminer.org/data/.
[4] http://dl.acm.org.
[5] https://snap.stanford.edu/data/higgs-twitter.html.
[6] https://www.flickr.com/.
[7] http://snap.stanford.edu/data/loc-gowalla.html.
[8] https://www.yelp.com/dataset/.
[9] http://jmcauley.ucsd.edu/data/amazon/index.html.
[10] http://qwone.com/jason/20Newsgroups/.
[11] http://www.mattmahoney.net/dc/textdata.

used text network datasets [22,26,28,58,113]. MovieLens[12], IMDB[13], MR[14], and YahooMusic[15] are widely used video and music datasets [21,26,28,36,39,42,47,52,82,88,91,115]. In the field of KGE, Wordnet[16], Freebase[17], and Yago[18] are widely used knowledge bases [21,42,97–103,105–109,111,113,119–123,126].

4.3 Open Source Code and Tools

Open source codes and platforms are also critical for HNE research. In this section, we list some open source codes and platforms.

4.3.1 Open Source Code

Open source code plays a very important role, enabling researchers to reproduce or improve existing HNE models. We extracted the open source code of some common HNE models from related papers, as listed in Table 9.

Table 9: Open source code for some typical HNE models

Model	Link	Language
MNE [41]	https://github.com/HKUST-KnowComp/MNE	python, C++
Metapath2vec [4]	https://ericdongyx.github.io/metapath2vec/m2v.html	C
HeteSpacey Walk [48]	https://github.com/HKUST-KnowComp/HeteSpaceyWalk	python
Hin2Vec [49]	https://github.com/csiesheep/hin2vec	python,C
Hyper-Gram [42]	https://github.com/HKUST-KnowComp/HPHG/	python
SHNE [50]	https://github.com/chuxuzhang/WSDM2019_SHNE	python
DIME [60]	http://www.ifmlab.org/files/code/Aligned-Autoencoder.zip	python
HGT [84]	https://github.com/acbull/pyHGT	python
HetGNN [80]	https://github.com/chuxuzhang/KDD2019_HetGNN	python
HAN [6]	https://github.com/Jhy1993/HAN	python
HAHE [73]	https://github.com/zhoushengisnoob/HAHE	python
Graph Inception [72]	https://github.com/zyz282994112/GraphInception.git	python
TransE [97]	https://everest.hds.utc.fr/doku.php?id=en:transe	python
ConvE [106]	https://github.com/TimDettmers/ConvE	python
HEER [111]	https://github.com/GentleZhu/HEER	python
KTUP [115]	https://github.com/TaoMiner/joint-kg-recommender	python
RELEARN [124]	https://github.com/yangji9181/RELEARN	python
GATNE [116]	https://github.com/THUDM/GATNE	python

[12] https://grouplens.org/datasets/movielens/.
[13] https://ai.stanford.edu/amaas/data/sentiment/.
[14] http://www.cs.cornell.edu/people/pabo/movie-review-data/.
[15] https://github.com/fmonti/mgcnn/tree/master/Data/yahoo_music.
[16] https://wordnet.princeton.edu/.
[17] https://developers.google.com/freebase.
[18] https://yago-knowledge.org/.

4.3.2 Open Source Platforms and Toolkits

Open source platforms and toolkits can help researchers quickly and easily build workflows for HNE. We summarize several popular toolkits and platforms for heterogeneous graphs as follows:

- Deep Graph Library (DGL). The DGL is an easy-to-use, high-performance, and scalable open source platform for deep learning on graph data. DGL collects rich example implementations of popular GNN models on a wide range of topics, such as GCMC, MAGNN, and HGT. It provides independent application programming interfaces (APIs) for homogeneous graphs, heterogeneous graphs, and knowledge graphs. The official website of DGL is https://www.dgl.ai/.
- PyTorch Geometric (PyG). PyG is a library for deep learning on irregularly structured input data such as graphs, point clouds and manifolds, built upon PyTorch. In addition to general graph data structures and processing methods, it contains a variety of recently published methods from the domains of relational learning and 3D data processing [137]. Related code and documentation can be found at https://pytorch-geometric.readthedocs.io/en/latest/.
- OpenKE. OpenKE is an open source framework for knowledge embedding organized by THUNLP based on the TensorFlow toolkit [138]. OpenKE provides a fast and stable toolkit including the most popular knowledge representation learning methods such as TansE, TransR, and TransD. OpenKE can support fast model verification and large-scale knowledge representation learning. Moreover, new models can be easily integrated into the OpenKE framework. Related toolkits and documentation are published at http://openke.thunlp.org/.
- OpenHINE. OpenHINE is an open source toolkit for HNE developed by researchers at the DMGroup of Beijing University of Posts and Telecommunications. It unifies the input/output/evaluation interface of the HNE model, and in addition, it revises and reproduces classic HNE models, including DHNE, HAN, HeGAN, HERec, HIN2vec, Metapath2vec, MetaGraph2vec, and RHINE. Related code and datasets can be found at https://github.com/BUPT-GAMMA/OpenHINE.

4.4 Performance Evaluation of Heterogeneous Network Embedding Models

In this subsection, we compare the performance of some typical HNE models on a subset of the publicly available DBLP dataset [139] for the link prediction task [112,125].

The dataset consists of 14,475 authors, 14,376 papers, 20 conferences, 8,920 words, and a total of 170,794 links. There are three types of relationships in the heterogeneous network: "paper-conference," "paper-author," and "paper-word." The HNE models in the comparison include examples from most of the six model types: PTE, metapath2vec, ESim, HIN2Vec, HGT, TransE, ConvE, HEER, RHINE, HGCN, and HeteHG-VAE. For the link prediction task for all models, the heterogeneous edges in the original HIN are divided into two parts: 80% of the edges were used for training, and the remaining 20% were used for testing. The performance evaluation metric was AUC. Table 10 presents the performance comparison of all models on the link prediction task.

Table 10: Performance comparison of some typical HNE models on link prediction task

Type	MF	RW			GNN	KGE			HB	
Model	PTE	MetaPath-2Vec	ESim	HIN-2Vec	HGT	TransE	ConvE	HEER	HGCN	HeteHG-VAE
AUC	0.846	0.796	0.8	0.80	0.862	0.783	0.833	0.829	0.858	0.898

As Table 10 reveals, the performances of the various HNE models on the link prediction task differ. The KGE model TransE performs relatively poorly because its model assumptions are too simple and it is only suitable for one-to-one semantic relations. Slightly better than the TransE model are the random walk-based models metapath2vec, Esim, and HIN2Vec. The performances of these three models are mediocre, indicating that the earlier random walk models have some disadvantages: ESim and HIN2Vec use random walk sequences that are too short; metapath2vec uses a single meta-path, which limits the amount of semantic information that can be captured. With performances above the performance of the RW-based models are two other KGE models (ConvE and HEER) and the matrix factorization-based model PTE. The GNN-based model HGT, hybrid model HGCN, and HeteHG-VAE perform relatively well, which demonstrates that GNN-based models capture the network information of HINs well. The hybrid model HeteHG-VAE uses the framework of variational AEs on top of the two-level GNN aggregation, which increases the robustness of the model. Hence, its performance is the best.

5 Future Directions and Open Issues

HNE has made great progress in recent years, which clearly shows that it is a powerful and promising graph analysis paradigm. In this section, we discuss and explore a range of open issues and possible future research directions (Fig. 14).

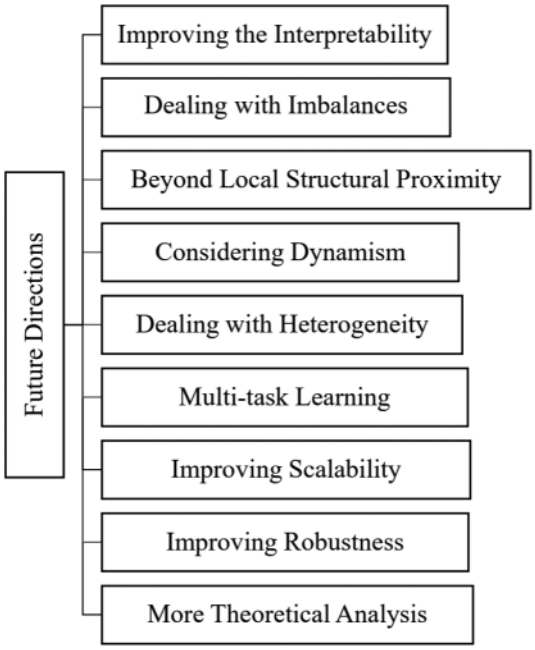


Figure 14: Future directions and open issues

5.1 Improving the Interpretability

Feature representation learning greatly reduces the workload of manual feature extraction, but such methods usually face the significant problem of poor interpretability, especially in some deep learning-based embedding models. The neural network structure is like a "black box." A common argument against deep neural networks is that their hidden weights and activations are often

unexplainable. Even if it has been experimentally verified that the embeddings generated by their own methods achieve good performance, one cannot gain a deep understanding of the application limitations of the model itself without interpreting the meaning of the learned feature representations. Generally, interpretability has two meanings: 1) interpretability for end users, i.e., explaining to users why the recommendation or prediction models based on HNEs give such results, and 2) interpretability for the implementer, i.e., enabling the researcher to understand the meanings of the weights, biases, and activation functions included in the model.

Some studies have considered the design of interpretable HNE models. Common approaches to improve model interpretability include exploiting the rich information in knowledge graphs to generate interpretable paths [133,140], attention mechanisms, and visual aids. We believe that it would be a good research direction to integrate inference models with ML techniques to design interpretable embedding models for specific ML tasks.

5.2 Dealing with Imbalances

Real-world HINs are often highly skewed. For such heterogeneous networks, if the skewness problem is not considered when designing the HNE model, it is likely that the model will be unbalanced during training, i.e., the model may be well-trained for some densely populated subnetworks but far from convergence in some relatively sparse subnetworks. A good feature representation learning model should overcome this problem.

A typical idea of existing studies that deal with skewness is to control the edge sampling to ensure the number of samples of different relation categories in the dataset is as balanced as possible [4,38,110]. However, this artificial control often changes the original distribution of the data, impacting the effectiveness of the embeddings generated by the model. We believe that future research requires a deeper understanding of the nature of skew phenomena and better solutions.

5.3 Beyond Local Structural Proximity

Many traditional graph representation learning models only consider the local structural proximity between nodes. Node embeddings generated in this type of model are usually based on "homogeneity". That is, the goal of solving the model is to make the feature representations of nodes with smaller network distances more similar, and vice versa more dissimilar. Most network structure information other than local network structure features is ignored, including structural role proximity, community structures, and motif structural features. This results in the generated node embeddings losing much topological information and they may not meet the needs of various specific ML tasks.

Currently, some studies based on homogeneous network embeddings consider the structural equivalence of nodes [37,141,142]. Moreover, some studies have considered the global structural features [70,77,143] or higher-order triangle motif [144] when generating network embeddings. However, the vast majority of existing HNE models ignore these issues. How future HNE models can improve the ability to capture structural features other than local structural proximity remains a challenging problem.

5.4 Considering Dynamism

Real-world networks are constantly changing. However, the vast majority of existing models are designed for static snapshot networks and do not consider the temporal characteristics of the network, resulting in the generated feature embeddings performing poorly on some time series-related ML tasks, such as abnormal event sequence detection and temporal link prediction. In addition, most existing

HNE models are transductive and can only generate embeddings for nodes that are currently observed in the network. This leads to the need to constantly retrain the HNE model when the network structure changes, which significantly increases resource consumption.

To handle the dynamics of the HINs, we consider there are two possible solutions: 1) when designing HNE models, consider the timing characteristics of the networks; 2) design inductive models such that they can generate embeddings for currently unobserved nodes in the HINs. There have been some studies devoted to generating embeddings for dynamic networks [145–149]. Such studies often use some deep memory networks, such as LSTM, to capture the temporal characteristics of network events. Moreover, some studies have designed inductive models to overcome the shortcomings of the transductive models [67,116]. The main feature of inductive models is that the input to the model does not depend on the number of nodes in the current networks, e.g., using auxiliary attributes other than the node one-hot encoding vectors as the input of the HNE models. Because of the dynamic and real-time nature of real-world networks, we believe that learning time-dependent embeddings for continuous-time dynamic networks and developing inductive network embedding models are very promising research directions.

5.5 Dealing with Heterogeneity

HINs usually contain rich heterogeneous attribute and relationship information: different types of nodes, different types of attributes, different types of semantic relations, and even the same type of node pairs may contain multiple different semantic relations. Highly heterogeneous networks bring challenges to the design of HNE models.

Of the existing models, GNN-based models usually have better ability to integrate the affiliated attributes. Meanwhile, KGE models perform better in handling various types of semantic relations. In addition, for multimodal heterogeneous networks containing multiple different attribute information (such as text, image, video), many studies have used various types of deep neural networks such as FNN, CNN or RNN to extract attribute features, and fuse the generated content embeddings into the final node embeddings [50,62,130]. This is more common in MF, RW, and AE types of HNE models. Some other studies have treated network node attributes as nodes in the network and generated embeddings for attribute nodes while generating entity node embeddings [84,112]. All of the above approaches have improved the ability of the HNE models to deal with the multimodal and multiplex issues. Owing to the inherent heterogeneity of complex networks, we believe that in the future research field of HNE, fully considering the heterogeneity is still worthwhile.

5.6 Multi-Task Learning

Most existing HNE models focus on generating feature representations for the nodes in the HINs, whereas some HNE models extend this task to generate edge embeddings, subgraph embeddings [136,150], etc. Meanwhile, some other models combine the network embedding task with specific downstream ML tasks to perform multiple tasks in one model simultaneously [30,31,58,61,63,76,118].

Compared with single-task learning, multi-task learning has the following advantages:

- it usually uses more data labels, which can better overcome the problem of data sparsity.
- it can effectively reduce overfitting of the model to specific task data and generate feature representations with better generalization performance due to the need to meet the training objectives of multiple tasks. For example, if edge and subgraph embeddings are generated simultaneously as a node embedding, the generated network embedding will be able to better capture the semantic information and global structural information contained in the network.

5.7 Improving Scalability

Many existing HNE models are designed for small-scale heterogeneous networks and cannot be scaled to large- and ultra-large-scale networks. However, real-world HINs often contain hundreds of millions of nodes and complex relationships. For such huge networks, in addition to improving the computing power of hardware computing platforms, some researchers have considered using a distributed parallelization strategy to divide large datasets into multiple small data samples in a "divide-and-conquer" manner [85]; there are also some studies that reduce the model complexity by data sampling to reduce the amount of data to be processed [151]. These methods effectively improve the scalability of the HNE model for large-scale networks. However, we believe that besides the above strategies, more and better methods to improve model efficiency and scalability are to be studied.

5.8 Improving Robustness

Available real-world heterogeneous network data is often incomplete, including incomplete nodes, link relationships, or affiliated information. Also, the available data often contains erroneous and noisy information.

A powerful HNE model needs to tolerate such data incompleteness and noise to learn more robust network embeddings. Generally speaking, generative models are more robust than traditional models due to adding noise to the model input [62,124] or using an adversarial learning framework [79,129]. More strategies to improve the robustness of HNE models are to be investigated.

5.9 More Theoretical Analysis

Most existing studies evaluate the performance of HNE models experimentally, and in-depth theoretical analysis is lacking, which may lead to model evaluation limitations and application bias problems. As mere experimental results may be limited to a specific dataset or task, failure to theoretically analyze and demonstrate the properties of the HNE model may result in the essence of the model not being deeply understood. In addition to the complexity analysis, if the theoretical characteristics of the model can be explored in more depth, it will substantially help us understand, apply, and expand the function of the model. For example, we could explore how deep learning models are intrinsically related to traditional models or explore higher-order Markov properties of random walks based on meta-paths. These theoretical analyses will bring more in-depth insights and rich perspectives to researchers.

6 Conclusions

In this paper, we provide a systematic and comprehensive review of research problems in HNE. From the encoder-decoder perspective, we divide existing HNE research into six categories: MF, RW, AE, GNN, KGE, and hybrid models. For each type of model, we first overview the basic common characteristics. Then, taking typical HNE models as examples, we systematically review each model type. We highlight the novel contribution of each representative HNE model, and summarize the advantages and disadvantages of each model type. We also provide a wealth of valuable relevant resources, including the application areas, benchmark datasets, open-source code and tools of this research field. Finally, we present an in-depth discussion about open issues and future directions. As a result, we believe that future HNE research must better deal with the heterogeneity, dynamics, skewness, and sparsity of the HINs; improve model interpretability, scalability, and robustness; and strengthen theoretical analysis to facilitate model application and expansion.

Acknowledgement: We would like to thank Dr. Rui Zhong for her kind suggestions on revisions to this paper.

Funding Statement: Our research work is supported by the National Key Research and Development Plan of China (2017YFB0503700, 2016YFB0501801); the National Natural Science Foundation of China (61170026, 62173157); the Thirteen Five-Year Research Planning Project of National Language Committee (No. YB135-149); the Fundamental Research Funds for the Central Universities (Nos. CCNU20QN022, CCNU20QN021, CCNU20ZT012).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Shi, C., Li, Y., Zhang, J., Sun, Y., Yu, P. S. (2016). A survey of heterogeneous information network analysis. *IEEE Transactions on Knowledge and Data Engineering, 29(1),* 17–37. https://doi.org/10.1109/TKDE.2016.2598561
2. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Proceedings of 27th Annual Conference on Neural Information Processing Systems 2013*, pp. 3111–3119. Nevada.
3. Levy, O., Goldberg, Y. (2014). Neural word embedding as implicit matrix factorization. *Proceedings of Annual Conference on Neural Information Processing Systems 2014*, pp. 2177–2185. Quebec.
4. Dong, Y., Chawla, N. V., Swami, A. (2017). metapath2vec: Scalable representation learning for heterogeneous networks. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 135–144. Halifax.
5. Huang, H., Song, Y., Ye, F., Xie, X., Shi, X. et al. (2021). Multi-stage network embedding for exploring heterogeneous edges. *ACM Transactions on Knowledge Discovery from Data, 15(1),* 1–27. https://doi.org/10.1145/3415157
6. Wang, X., Ji, H., Shi, C., Wang, B., Ye, Y. et al. (2019). Heterogeneous graph attention network. *Proceedings of the World Wide Web Conference*, pp. 2022–2032. San Francisco.
7. Peng, C., Wang, X., Pei, J., Zhu, W. (2019). A survey on network embedding. *IEEE Transactions on Knowledge and Data Engineering, 31(5),* 833–852. https://doi.org/10.1109/TKDE.2018.2849727
8. Hamilton, W. L., Ying, R., Leskovec, J. (2017). Representation learning on graphs: Methods and applications. *IEEE Data Engineering Bulletin, 40(3),* 52–74.
9. Cai, H., Zheng, V. W., Chang, C. C. (2018). A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering, 30(9),* 1616–1637. https://doi.org/10.1109/TKDE.2018.2807452
10. Goyal, P., Ferrara, E. (2018). Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems, 151(1),* 78–94. https://doi.org/10.1016/j.knosys.2018.03.022
11. Wang, Y., Yao, Y., Tong, H., Xu, F., Lu, J. (2019). A brief review of network embedding. *Big Data Mining and Analytics, 2(1),* 35–47. https://doi.org/10.26599/BDMA.2018.9020029
12. Zhang, D., Jie, Y., Zhu, X., Zhang, C. (2020). Network representation learning: A survey. *IEEE Transactions on Big Data, 6(1),* 3–28. https://doi.org/10.1109/TBDATA.2018.2850013
13. Li, B., Pi, D. (2020). Network representation learning: A systematic literature review. *Neural Computing and Applications, 32(21),* 16647–16679. https://doi.org/10.1007/s00521-020-04908-5
14. Hou, M., Ren, J., Zhang, D., Kong, X., Xia, F. (2020). Network embedding: Taxonomies, frameworks and applications. *Computer Science Review, 38(99),* 100296. https://doi.org/10.1016/j.cosrev.2020.100296

15. Yang, C., Xiao, Y., Zhang, Y., Sun, Y., Han, J. (2022). Heterogeneous network representation learning: A unified framework with survey and benchmark. *IEEE Transactions on Knowledge and Data Engineering, 34(10),* 4854–4873. https://doi.org/10.1109/TKDE.2020.3045924
16. Xie, Y., Yu, B., Lv, S., Zhang, C., Gong, M. (2021). A survey on heterogeneous network representation learning. *Pattern Recognition, 116(7),* 107936. https://doi.org/10.1016/j.patcog.2021.107936
17. Dong, Y., Hu, Z., Wang, K., Sun, Y., Tang, J. (2020). Heterogeneous network representation learning. *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pp. 4861–4867. Yokohama.
18. Wang, X., Bo, D., Shi, C., Fan, S., Ye, Y. et al. (2020). A survey on heterogeneous graph embedding: methods, techniques, applications and sources. arXiv:2011.14867v1.
19. Ji, F., Zhao, Z., Zhou, H., Chi, H., Li, C. (2020). A comparative study on heterogeneous information network embeddings. *Journal of Intelligent & Fuzzy Systems: Applications in Engineering and Technology, 39(3),* 3463–3473. https://doi.org/10.3233/JIFS-191796
20. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J. et al. (2015). LINE: Large-scale information network embedding. *Proceedings of the 24th International Conference on World Wide Web*, pp. 1067–1077. Florence.
21. Tu, K., Cui, P., Wang, X., Wang, F., Zhu, W. (2018). Structural deep embedding for hyper-networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 426–433. New Orleans.
22. Zhao, Y., Liu, Z., Sun, M. (2015). Representation learning for measuring entity relatedness with rich information. *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, pp. 1412–1418. Buenos Aires.
23. Ahmed, A., Shervashidze, N., Narayanamurthy, S. M., Josifovski, V., Smola, A. J. (2013). Distributed large-scale natural graph factorization. *Proceedings of 22nd International World Wide Web Conference*, pp. 37–48. Rio de Janeiro.
24. Belkin, M., Niyogi, P. (2001). Laplacian eigenmaps and spectral techniques for embedding and clustering. *Proceedings of Annual Conference on Neural Information Processing Systems*, pp. 585–591. Vancouver.
25. Ou, M., Cui, P., Pei, J., Zhang, Z., Zhu, W. (2016). Asymmetric transitivity preserving graph embedding. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1105–1114. San Francisco.
26. Tang, J., Qu, M., Mei, Q. (2015). PTE: predictive text embedding through large-scale heterogeneous text networks. *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1165–1174. Sydney.
27. Su, Y., Li, X., Zha, D., Tang, W., Jiang, Y. et al. (2019). Hrec: Heterogeneous graph embedding-based personalized point-of-interest recommendation. *Proceedings of the 26th International Conference on ICONIP 2019*, pp. 37–49. Sydney.
28. Tang, J., Qu, M., Mei, Q. (2016). Identity-sensitive word embedding through heterogeneous networks. arXiv:1611.09878v1.
29. Wang, Y., Feng, C., Chen, L., Yin, H., Guo, C. et al. (2019). User identitylinkage across social networks via linked heterogeneous network embedding. *World Wide Web, 22(6),* 2611–2632.
30. Qu, M., Tang, J., Shang, J., Ren, X., Zhang, M. et al. (2017). An attention-based collaboration framework for multi-view network representation learning. *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 1767–1776. Singapore.
31. Hu, J., Qian, S., Fang, Q., Liu, X., Xu, C. (2019). A^2CMHNE: Attention-aware collaborative multimodal heterogeneous network embedding. *ACM Transactions on Multimedia Computing Communications and Applications, 15(2),* 1–17.
32. Bl, A., Dpa, B., Yl, A., Iak, A., Lin, C. C. (2020). Multi-source information fusion based heterogeneous network embedding. *Information Sciences, 534(1),* 53–71. https://doi.org/10.1016/j.ins.2020.05.012

33. Perozzi, B., Al-Rfou, R., Skiena, S. (2014). Deepwalk: Online learning of social representations. *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 701–710. New York.
34. Zhu, Y., Guan, Z., Tan, S., Liu, H., Cai, D. et al. (2016). Heterogeneous hypergraph embedding for document recommendation. *Neurocomputing, 216,* 150–162.
35. Sun, L., He, L., Huang, Z., Cao, B., Xia, C. et al. (2018). Joint embedding of meta-path and meta-graph for heterogeneous information networks. *Proceedings of the 2018 IEEE International Conference on Big Knowledge*, pp. 131–138. Singapore.
36. Zheng, S., Guan, D., Yuan, W. (2022). Semantic-aware heterogeneous information network embedding with incompatible meta-paths. *World Wide Web-internet and Web Information Systems, 25(1),* 1–21.
37. Grover, A., Leskovec, J. (2016). node2vec: Scalable feature learning for networks. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 855–864. San Francisco.
38. Hussein, R., Yang, D., Cudré-Mauroux, P. (2018). Are meta-paths necessary? Revisiting heterogeneous graph embeddings. *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pp. 437–446. Torino.
39. Jiang, J., Li, Z., Ju, C. J., Wang, W. (2020). MARU: meta-context aware random walks for heterogeneous network representation learning. *Proceedings of The 29th ACM International Conference on Information and Knowledge Management*, pp. 575–584. Virtual Event, Ireland.
40. Cai, X., Han, J., Pan, S., Yang, L. (2018). Heterogeneous information network embedding based personalized query-focused astronomy reference paper recommendation. *International Journal of Computational Intelligence Systems, 11(1),* 591–599. https://doi.org/10.2991/ijcis.11.1.44
41. Zhang, H., Qiu, L., Yi, L., Song, Y. (2018). Scalable multiplex network embedding. *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pp. 3082–3088. Stockholm.
42. Huang, J., Liu, X., Song, Y. (2019). Hyper-path-based representation learning for hyper-networks. *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 449–458. Beijing.
43. Jiang, Z., Gao, Z., Lan, J., Yang, H., Lu, Y. et al. (2020). Task-oriented genetic activation for large-scale complex heterogeneous graph embedding. *Proceedings of the Web Conference 2020*, pp. 1581–1591. Taipei.
44. Sun, Y., Han, J., Yan, X., Yu, P. S., Wu, T. (2011). PathSim: Meta path-based top-k similarity search in heterogeneous information networks. *Proceedings of the VLDB Endowment, 4(11),* 992–1003. https://doi.org/10.14778/3402707.3402736
45. Zhang, D., Yin, J., Zhu, X., Zhang, C. (2018). Metagraph2vec: Complex semantic path augmented heterogeneous network embedding. *Proceedings of 22nd Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 196–208. Melbourne.
46. Shi, C., Hu, B., Zhao, X., Yu, P. (2019). Heterogeneous information network embedding for recommendation. *IEEE Transactions on Knowledge and Data Engineering, 31(2),* 357–370. https://doi.org/10.1109/TKDE.2018.2833443
47. Wang, Z., Liu, H., Du, Y., Wu, Z., Zhang, X. (2019). Unified embedding model over heterogeneous information network for personalized recommendation. *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pp. 3813–3819. Macao.
48. He, Y., Song, Y., Li, J., Ji, C., Peng, J. et al. (2019). Hetespaceywalk: A heterogeneous spacey random walk for heterogeneous information network embedding. *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 639–648. Beijing.
49. Fu, T., Lee, W., Lei, Z. (2017). Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning. *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 1797–1806. Singapore.

50. Zhang, C., Swami, A., Chawla, N. V. (2019). SHNE: Representation learning for semantic-associated heterogeneous networks. *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pp. 690–698. Melbourne.
51. Shang, J., Meng, Q., Liu, J., Kaplan, L. M., Jian, P. (2016). Meta-path guided embedding for similarity search in large-scale heterogeneous information networks. arXiv:1610.09769v1.
52. Zhang, Y., Wang, X., Liu, N., Shi, C. (2022). Embedding heterogeneous information network in hyperbolic spaces. *ACM Transactions on Knowledge Discovery from Data, 16(2),* 1–23. https://doi.org/10.1145/3468674
53. Chen, T., Sun, Y. (2017). Task-guided and path-augmented heterogeneous network embedding for author identification. *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pp. 295–304. Cambridge.
54. Yang, C., Liu, Z., Zhao, D., Sun, M., Chang, E. Y. (2015). Network representation learning with rich text information. *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, pp. 2111–2117. Buenos Aires.
55. Wang, D., Cui, P., Zhu, W. (2016). Structural deep network embedding. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1225–1234. San Francisco.
56. Cao, S., Lu, W., Xu, Q. (2016). Deep neural networks for learning graph representations. *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pp. 1145–1152. Phoenix.
57. Kipf, T. N., Welling, M. (2016). Variational graph auto-encoders. arXiv:1611.07308v1.
58. Wang, H., Zhang, F., Hou, M., Xie, X., Guo, M. et al. (2018). SHINE: Signed heterogeneous information network embedding for sentiment link prediction. *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pp. 592–600. Marina Del Rey.
59. Fu, G., Yuan, B., Duan, Q., Yao, X. (2019). Representation learning for heterogeneous information networks via embedding events. *Proceedings of the 26th International Conference on Neural Information Processing*, pp. 327–339. Sydney.
60. Zhang, J., Xia, C., Zhang, C., Cui, L., Fu, Y. et al. (2017). BL-MNE: Emerging heterogeneous social network embedding through broad learning with aligned autoencoder. *Proceedings of the 2017 IEEE International Conference on Data Mining*, pp. 605–614. New Orleans.
61. Ji, H., Shi, C., Wang, B. (2018). Attention based meta path fusion for heterogeneous information network embedding. *Proceedings of the 15th Pacific Rim International Conference on Artificial Intelligence*, pp. 348–360. Nanjing.
62. Huang, F., Zhang, X., Li, C., Li, Z., He, Y. et al. (2018). Multimodal network embedding via attention based multi-view variational autoencoder. *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*, pp. 108–116. Yokohama.
63. Fan, S., Shi, C., Wang, X. (2018). Abnormal event detection via heterogeneous information network embedding. *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pp. 1483–1486. Torino.
64. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C. et al. (2021). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems, 32(1),* 4–24. https://doi.org/10.1109/TNNLS.2020.2978386
65. Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C. et al. (2020). Graph neural networks: A review of methods and applications. *AI Open, 1(1),* 57–81. https://doi.org/10.1016/j.aiopen.2021.01.001
66. Kipf, T. N., Welling, M. (2017). Semi-supervised classification with graph convolutional networks. *Proceedings of the 5th International Conference on Learning Representations*, Toulon.
67. Hamilton, W. L., Ying, Z., Leskovec, J. (2017). Inductive representation learning on large graphs. *Proceedings of the Annual Conference on Neural Information Processing Systems 2017*, pp. 1024–1034. Long Beach, CA.

68. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P. et al. (2018). Graph attention networks. *Proceedings of the 6th International Conference on Learning Representations*, Vancouver.
69. Li, Y., Tarlow, D., Brockschmidt, M., Zemel, R. S. (2016). Gated graph sequence neural networks. *Proceedings of the 4th International Conference on Learning Representations*, San Juan.
70. Park, C., Kim, D., Han, J., Yu, H. (2020). Unsupervised attributed multiplex network embedding. *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*, pp. 5371–5378. New York.
71. Velickovic, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y. et al. (2019). Deep graph infomax. *Proceedings of the 7th International Conference on Learning Representations*, New Orleans.
72. Zhang, Y., Xiong, Y., Kong, X., Li, S., Mi, J. et al. (2018). Deep collective classification in heterogeneous information networks. *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pp. 399–408. Lyon.
73. Zhou, S., Bu, J., Wang, X., Chen, J., Hu, B. et al. (2019). HAHE: Hierarchical attentive heterogeneous information network embedding. arXiv:1902.01475v2.
74. Zhang, Y., Fan, Y., Ye, Y., Zhao, L., Shi, C. (2019). Key player identification in underground forums over attributed heterogeneous information network embedding framework. *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 549–558. Beijing.
75. Fan, S., Zhu, J., Han, X., Shi, C., Hu, L. et al. (2019). Metapath-guided heterogeneous graph neural network for intent recommendation. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2478–2486. Anchorage.
76. Fu, X., Zhang, J., Meng, Z., King, I. (2020). MAGNN: Metapath aggregated graph neural network for heterogeneous graph embedding. *Proceedings of the Web Conference 2020*, pp. 2331–2341. Taipei.
77. Ren, Y., Liu, B., Huang, C., Dai, P., Bo, L. et al. (2019). Heterogeneous deep graph infomax. arXiv:1911.08538v5.
78. Sun, X., Yin, H., Liu, B., Chen, H., Cao, J. et al. (2021). Heterogeneous hypergraph embedding for graph classification. *Proceedings of the Fourteenth ACM International Conference on Web Search and Data Mining*, pp. 725–733. Virtual Event, Israel.
79. Zhang, M., Wang, X., Zhu, M., Shi, C., Zhang, Z. et al. (2022). Robust heterogeneous graph neural networks against adversarial attacks. *Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence*, pp. 4363–4370. Virtual Event.
80. Zhang, C., Song, D., Huang, C., Swami, A., Chawla, N. V. (2019). Heterogeneous graph neural network. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 793–803. Anchorage.
81. Zheng, V. W., Sha, M., Li, Y., Yang, H., Fang, Y. et al. (2018). Heterogeneous embedding propagation for large-scale e-commerce user alignment. *Proceedings of the IEEE International Conference on Data Mining*, pp. 1434–1439. Singapore.
82. Chen, X., Yu, G., Wang, J., Domeniconi, C., Li, Z. et al. (2019). ActiveHNE: Active heterogeneous network embedding. *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pp. 2123–2129. Macao.
83. Chen, W., Gu, Y., Ren, Z., He, X., Xie, H. et al. (2019). Semi-supervised user profiling with heterogeneous graph attention networks. *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pp. 2116–2122. Macao.
84. Hu, Z., Dong, Y., Wang, K., Sun, Y. (2020). Heterogeneous graph transformer. *Proceedings of the Web Conference 2020*, pp. 2704–2710. Taipei.
85. Imran, M., Yin, H., Chen, T., Huang, Z., Zheng, K. (2022). DeHIN: A decentralized framework for embedding large-scale heterogeneous information networks. arXiv:2201.02757v1.
86. Jing, B., Park, C., Tong, H. (2021). HDMI: High-order deep multiplex infomax. *Proceedings of the Web Conference 2021*, pp. 2414–2424. Virtual Event, Ljubljana.

87. Liu, Z., Huang, C., Yu, Y., Fan, B., Dong, J. (2020). Fast attributed multiplex heterogeneous network embedding. *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*, pp. 995–1004. Virtual Event, Ireland.
88. Wang, Y., Tang, S., Lei, Y., Song, W., Wang, S. et al. (2020). Disenhan: Disentangled heterogeneous graph attention network for recommendation. *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*, pp. 1605–1614. Virtual Event, Ireland.
89. Wang, P., Agarwal, K., Ham, C., Choudhury, S., Reddy, C. K. (2021). Self-supervised learning of contextual embeddings for link prediction in heterogeneous networks. *Proceedings of the Web Conference 2021*, pp. 2946–2957. Virtual Event, Ljubljana.
90. Chang, Y., Chen, C., Hu, W., Zheng, Z., Zhou, X. et al. (2022). Megnn: Meta-path extracted graph neural network for heterogeneous graph representation learning. *Knowledge-Based Systems, 235(1),* 107611. https://doi.org/10.1016/j.knosys.2021.107611
91. Berg, R., Kipf, T. N., Welling, M. (2017). Graph convolutional matrix completion. arXiv:1706.02263v2.
92. Li, Q., Han, Z., Wu, X. (2018). Deeper insights into graph convolutional networks for semi-supervised learning. *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pp. 3538–3545. New Orleans.
93. Miller, George, A. (1995). Wordnet: A lexical database for english. *Communications of the ACM, 38(11),* 39–41. https://doi.org/10.1145/219717.219748
94. Bollacker, K. D., Evans, C., Paritosh, P., Sturge, T., Taylor, J. (2008). Freebase: A collaboratively created graph database for structuring human knowledge. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 1247–1250. Vancouver.
95. Suchanek, F. M., Kasneci, G., Weikum, G. (2007). Yago: A core of semantic knowledge. *Proceedings of the 16th International Conference on World Wide Web*, pp. 697–706. Banff, Alberta.
96. Wang, Q., Mao, Z., Wang, B., Guo, L. (2017). Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering, 29(12),* 2724–2743. https://doi.org/10.1109/TKDE.2017.2754499
97. Bordes, A., Usunier, N., García-Durán, A., Weston, J., Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. *Proceedings of the 27th Annual Conference on Neural Information Processing Systems*, pp. 2787–2795. Nevada.
98. Wang, Z., Zhang, J., Feng, J., Chen, Z. (2014). Knowledge graph embedding by translating on hyperplanes. *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pp. 1112–1119. Quebec City.
99. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X. (2015). Learning entity and relation embeddings for knowledge graph completion. *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pp. 2181–2187. Texas.
100. Ji, G., He, S., Xu, L., Liu, K., Zhao, J. (2015). Knowledge graph embedding via dynamic mapping matrix. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pp. 687–696. Beijing.
101. He, S., Liu, K., Ji, G., Zhao, J. (2015). Learning to represent knowledge graphs with gaussian embedding. *Proceedings of the 24th ACM International Conference on Information and Knowledge Management*, pp. 623–632. Melbourne.
102. Xiao, H., Huang, M., Zhu, X. (2016). Transg: A generative model for knowledge graph embedding. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pp. 2316–2325. Berlin.
103. Bordes, A., Weston, J., Collobert, R., Bengio, Y. (2011). Learning structured embeddings of knowledge bases. *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, San Francisco.
104. Nickel, M., Tresp, V., Kriegel, H. (2011). A three-way model for collective learning on multi-relational data. *Proceedings of the 28th International Conference on Machine Learning*, pp. 809–816. Bellevue.

105. Yang, B., Yih, W., He, X., Gao, J., Deng, L. (2015). Embedding entities and relations for learning and inference in knowledge bases. *Proceedings of the 3rd International Conference on Learning Representations*, San Diego.

106. Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S. (2018). Convolutional 2D knowledge graph embeddings. *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pp. 1811–1818. New Orleans.

107. Socher, R., Chen, D., Manning, C. D., Ng, A. Y. (2013). Reasoning with neural tensor networks for knowledge base completion. *Proceedings of the 27th Annual Conference on Neural Information Processing Systems*, pp. 926–934. Nevada.

108. Sun, Z., Deng, Z., Nie, J., Tang, J. (2019). Rotate: Knowledge graph embedding by relational rotation in complex space. *Proceedings of the 7th International Conference on Learning Representations*, New Orleans.

109. Nickel, M., Rosasco, L., Poggio, T. A. (2016). Holographic embeddings of knowledge graphs. *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pp. 1955–1961. Phoenix.

110. Chen, H., Yin, H., Wang, W., Wang, H., Nguyen, Q. V. H. et al. (2018). PME: Projected metric embedding on heterogeneous networks for link prediction. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1177–1186. London.

111. Shi, Y., Zhu, Q., Guo, F., Zhang, C., Han, J. (2018). Easing embedding learning by comprehensive transcription of heterogeneous information networks. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2190–2199. London.

112. Lu, Y., Shi, C., Hu, L., Liu, Z. (2019). Relation structure-aware heterogeneous information network embedding. *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, pp. 4456–4463. Honolulu.

113. Feng, M., Hsu, C., Li, C., Yeh, M., Lin, S. (2019). MARINE: Multi-relational network embeddings with relational proximity and node attributes. *Proceedings of the World Wide Web Conference*, pp. 470–479. San Francisco.

114. Lin, Y., Liu, Z., Luan, H., Sun, M., Rao, S. et al. (2015). Modeling relation paths for representation learning of knowledge bases. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 705–714. Lisbon.

115. Cao, Y., Wang, X., He, X., Hu, Z., Chua, T. (2019). Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. *Proceedings of the World Wide Web Conference*, pp. 151–161. San Francisco.

116. Cen, Y., Zou, X., Zhang, J., Yang, H., Zhou, J. et al. (2019). Representation learning for attributed multiplex heterogeneous network. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1358–1368. Anchorage.

117. Qiao, Z., Du, Y., Fu, Y., Wang, P., Zhou, Y. (2019). Unsupervised author disambiguation using heterogeneous graph convolutional network embedding. *Proceedings of the 2019 IEEE International Conference on Big Data (Big Data)*, pp. 910–919. Los Angeles.

118. Trivedi, R., Sisman, B., Dong, X. L., Faloutsos, C., Ma, J. et al. (2018). LinkNBed: Multi-graph representation learning with entity linkage. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pp. 252–262. Melbourne.

119. Shang, C., Tang, Y., Huang, J., Bi, J., He, X. et al. (2019). End-to-end structure-aware convolutional networks for knowledge base completion. *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, pp. 3060–3067. Honolulu.

120. Wang, K., Liu, Y., Xu, X., Lin, D. (2018). Knowledge graph embedding with entity neighbors and deep memory network. arXiv:1808.03752v1.

121. Oh, B., Seo, S., Lee, K. (2018). Knowledge graph completion by context-aware convolutional learning with multi-hop neighborhoods. *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pp. 257–266. Torino.

122. Ning, Z., Qiao, Z., Dong, H., Du, Y., Zhou, Y. (2021). Lightcake: A lightweight framework for context-aware knowledge graph embedding. *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 181–193. Virtual Event.
123. Li, Z., Liu, H., Zhang, Z., Liu, T., Xiong, N. N. (2022). Learning knowledge graph embedding with heterogeneous relation attention networks. *IEEE Transactions on Neural Networks and Learning Systems, 33(8),* 3961–3973. https://doi.org/10.1109/TNNLS.2021.3055147
124. Yang, C., Zhang, J., Wang, H., Li, S., Kim, M. et al. (2020). Relation learning on social networks with multi-modal graph edge variational autoencoders. *Proceedings of the Thirteenth ACM International Conference on Web Search and Data Mining*, pp. 699–707. Houston.
125. Fan, H., Zhang, F., Wei, Y., Li, Z., Dai, Q. (2022). Heterogeneous hypergraph variational autoencoder for link prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 44(8),* 4125–4138.
126. Chen, D., Li, Y., Ding, B., Shen, Y. (2020). An adaptive embedding framework for heterogeneous information networks. *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pp. 165–174. Virtual Event, Ireland.
127. Zhong, Z., Li, C. T., Pang, J. (2020). Reinforcement learning enhanced heterogeneous graph neural network. arXiv:2010.13735.
128. Zhao, K., Bai, T., Wu, B., Wang, B., Zhang, Y. et al. (2020). Deep adversarial completion for sparse heterogeneous information network embedding. *Proceedings of The Web Conference 2020*, pp. 508–518. Taipei.
129. Hu, B., Fang, Y., Shi, C. (2019). Adversarial learning on heterogeneous information networks. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 120–129. Anchorage.
130. Chu, Y., Feng, C., Guo, C. (2018). Social-guided representation learning for images via deep heterogeneous hypergraph embedding. *Proceedings of the 2018 IEEE International Conference on Multimedia and Expo*, pp. 1–6. San Diego.
131. Xu, L., Wei, X., Cao, J., Yu, P. S. (2017). Embedding of embedding (EOE), joint embedding for coupled heterogeneous networks. *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pp. 741–749. Cambridge.
132. Fu, Y., Xiong, Y., Yu, P. S., Tao, T., Zhu, Y. (2019). Metapath enhanced graph attention encoder for hins representation learning. *Proceedings of the 2019 IEEE International Conference on Big Data (Big Data)*, pp. 1103–1110. Los Angeles.
133. Ai, Q., Azizi, V., Chen, X., Zhang, Y. (2018). Learning heterogeneous knowledge base embeddings for explainable recommendation. *Algorithms, 11(9),* 137. https://doi.org/10.3390/a11090137
134. Schlichtkrull, M. S., Kipf, T. N., Bloem, P., van den Berg, R., Titov, I. et al. (2018). Modeling relational data with graph convolutional networks. *Proceedings of the 15th International Conference on the Semantic Web*, pp. 593–607. Heraklion.
135. Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L. et al. (2008). Arnetminer: Extraction and mining of academic social networks. *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 990–998. Las Vegas.
136. Du, B., Tong, H. (2019). MrMine: Multi-resolution multi-network embedding. *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 479–488. Beijing.
137. Fey, M., Lenssen, J. E. (2019). Fast graph representation learning with pytorch geometric. arXiv:1903.02428v3.
138. Han, X., Cao, S., Lv, X., Lin, Y., Liu, Z. et al. (2018). OpenKE: An open toolkit for knowledge embedding. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 139–144. Brussels.

139. Ji, M., Sun, Y., Danilevsky, M., Han, J., Gao, J. (2010). Graph regularized transductive classification on heterogeneous information networks. *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 570–586. Barcelona.

140. Chen, H., Li, Y., Sun, X., Xu, G., Yin, H. (2021). Temporal meta-path guided explainable recommendation. *Proceedings of the Fourteenth ACM International Conference on Web Search and Data Mining*, pp. 1056–1064. Virtual Event, Israel.

141. Ribeiro, L. F. R., Saverese, P. H. P., Figueiredo, D. R. (2017). *struc2vec:* Learning node representations from structural identity. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 385–394. Halifax.

142. Donnat, C., Zitnik, M., Hallac, D., Leskovec, J. (2018). Learning structural node embeddings via diffusion wavelets. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1320–1329. London.

143. Han, X., Jiang, Z., Liu, N., Song, Q., Li, J. et al. (2022). Geometric graph representation learning via maximizing rate reduction. *Proceedings of the ACM Web Conference 2022*, pp. 1226–1237. Virtual Event, Lyon.

144. Hu, Q., Lin, F., Wang, B., Li, C. (2022). MBRep: Motif-based representation learning in heterogeneous networks. *Expert Systems with Application, 190(6295),* 116031. https://doi.org/10.1016/j.eswa.2021.116031

145. Yin, Y., Ji, L. X., Zhang, J. P., Pei, Y. L. (2019). DHNE: Network representation learning method for dynamic heterogeneous networks. *IEEE Access, 7,* 134782–134792. https://doi.org/10.1109/ACCESS.2019.2942221

146. Xue, H., Yang, L., Jiang, W., Wei, Y., Hu, Y. et al. (2020). Modeling dynamic heterogeneous network for link prediction using hierarchical attention with temporal RNN. *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 282–298. Ghent.

147. Luo, W., Zhang, H., Yang, X., Bo, L., Yang, X. et al. (2020). Dynamic heterogeneous graph neural network for real-time event prediction. *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3213–3223. Virtual Event.

148. Wang, X., Lu, Y., Shi, C., Wang, R., Mou, S. (2022). Dynamic heterogeneous information network embedding with meta-path based proximity. *IEEE Transactions on Knowledge and Data Engineering, 34(3),* 1117–1132. https://doi.org/10.1109/TKDE.2020.2993870

149. Xie, Y., Ou, Z., Chen, L., Liu, Y., Xu, K. et al. (2021). Learning and updating node embedding on dynamic heterogeneous information network. *Proceedings of The Fourteenth ACM International Conference on Web Search and Data Mining*, pp. 184–192. Virtual Event, Israel.

150. Narayanan, A., Chandramohan, M., Venkatesan, R., Chen, L., Jaiswal, S. (2017). graph2vec: Learning distributed representations of graphs. arXiv:1707.05005v1.

151. Yang, Z., Ding, M., Zhou, C., Yang, H., Zhou, J. et al. (2020). Understanding negative sampling in graph representation learning. *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1666–1676. Virtual Event, CA.