Heterogeneous Network Representation Learning: Survey and Benchmark

Carl Yang*, Yuxin Xiao*, Yu Zhang*, Yizhou Sun, *Member, IEEE,* and Jiawei Han, *Life Fellow, IEEE*

Abstract—Since real-world objects and their interactions are often multi-modal and multi-typed, heterogeneous networks have been widely used as a more powerful, realistic, and generic superclass of traditional homogeneous networks (graphs). Meanwhile, representation learning (*a.k.a.* embedding) has recently been intensively studied and shown effective for various network mining and analytical tasks. In this work, we aim to provide a unified framework to deeply summarize and evaluate existing research on heterogeneous network embedding (HNE), which includes but goes beyond a normal survey. Since there has already been a broad body of HNE algorithms, as the first contribution of this work, we provide a generic paradigm for the systematic categorization and analysis over the merits of various existing HNE algorithms. Moreover, existing HNE algorithms, though mostly claimed generic, are often evaluated on different datasets. Understandable due to the application favor of HNE, such indirect comparisons largely hinder the proper attribution of improved task performance towards effective data preprocessing and novel technical design, especially considering the various ways possible to construct a heterogeneous network from real-world application data. Therefore, as the second contribution, we create four benchmark datasets with various properties regarding scale, structure, attribute/label availability, and *etc*. from different sources, towards handy and fair evaluations of HNE algorithms. As the third contribution, we carefully refactor and amend the implementations and create friendly interfaces for eleven popular HNE algorithms, and provide all-around comparisons among them over multiple tasks and experimental settings.

By putting all existing HNE algorithms under a unified framework, we aim to provide a universal reference and guideline for the understanding and development of HNE algorithms. Meanwhile, by open-sourcing all data and code, we envision to serve the community with an ready-to-use benchmark platform to test and compare the performance of existing and future HNE algorithms (https://github.com/yangji9181/HNE).

Index Terms—heterogeneous network, representation learning, survey, benchmark

1 Introduction

Networks and graphs constitute a canonical and ubiquitous paradigm for the modeling of interactive objects, which has drawn significant research attention from various scientific domains [1], [2], [3], [4], [5], [6]. However, real-world objects and interactions are often multi-modal and multi-typed (*e.g.*, authors, papers, venues and terms in a publication network [7], [8]; users, places, categories and GPS-coordinates in a location-based social network [9], [10], [11]; and genes, proteins, diseases and species in a biomedical network [12], [13]). To capture and exploit such node and link heterogeneity, heterogeneous networks have been proposed and widely used in many real-world network mining scenarios, such as meta-path based similarity search [14], [15], [16], node classification and clustering [17], [18], [19], knowledge base completion [20], [21], [22], and recommendations [23], [24], [25].

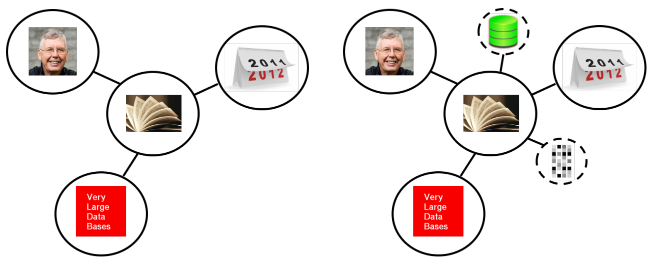
In the meantime, current research on graphs has largely focused on representation learning (embedding), especially following the pioneer of neural network based algorithms that demonstrate revealing empirical evidence towards unprecedentedly effective yet efficient graph mining [26], [27], [28]. They aim to convert graph data (*e.g.*, nodes [29], [30], [31], [32], [33], [34], [35], [36], links [37], [38], [39], [40], and subgraphs [41], [42], [43], [44]) into low dimensional distributed vectors in the embedding space where the graph topological information (*e.g.*, higher-order proximity [45], [46], [47], [48] and structure [49], [50], [51], [52]) is preserved. Such *embedding vectors* are then directly executable by various downstream machine learning algorithms [53], [54], [55].

Right on the intersection of heterogeneous networks and graph embedding, heterogeneous network embedding (HNE) recently has also received significant research attention [56], [57], [58], [59], [60], [61], [62], [63], [64], [65], [66], [67], [68], [69], [70], [71], [72], [73], [74], [75], [76]. Due to the application favor of HNE, many algorithms have been separately developed in different application domains such as search and recommendations [5], [23], [77], [78]. Moreover, as knowledge bases (KBs) also fall under the general umbrella of heterogeneous networks, many KB embedding algorithms can be compared with the HNE ones [4], [20], [21], [79], [80], [81], [82], [83], [84].

Unfortunately, various HNE algorithms are developed in quite disparate communities across academia and industry. They have never been systematically and comprehensively analyzed either in concepts or through experiments. In fact, due to the lack of benchmark platforms (with ready-to-use datasets and baselines), researchers often tend to construct their own datasets and re-implement a few most popular (sometimes outdated) algorithms for comparison, which renders fair performance evaluation and clear improvement

*Three authors contribute equally.

• *Carl Yang is with Emory University; Yuxin Xiao, Yu Zhang and Jiawei Han are with University of Illinois, Urbana Champaign; Yizhou Sun is with University of California, Los Angeles.*



(a) Plain heterogeneous net. (b) Labeled attributed net.

Fig. 1: **Different heterogeneous networks constructed from the same real-world application data.**

attribution extremely hard, if not impossible.

Simply consider the toy examples of a publication dataset in Figure 1.[1] Earlier HNE algorithms like metapath2vec [59] were developed on the heterogeneous network with node types of authors, papers and venues as in (a). However, one can enrich papers with a large number of terms and topics as additional nodes as in (b), which makes the random-walk based shallow embedding algorithms rather ineffective, but favors neighborhood aggregation based deep graph neural networks like R-GCN [67]. Moreover, one can further include node attributes like term embedding and labels like research fields and make them only available to the semi-supervised inductive learning algorithms, which may introduce even more bias [66], [75], [85], [86]. Eventually, it is often hard to clearly attribute performance gains between technical novelty and data tweaking.

In this work, we first formulate a unified yet flexible mathematical paradigm of HNE algorithms, easing the understanding of the critical merits of each model (Section 2). Particularly, based on a uniform taxonomy that clearly categorizes and summarizes the existing models (and likely future models), we propose a generic objective function of *network smoothness*, and reformulate all existing models into this uniform paradigm while highlighting their invididual novel contributions (Section 3). We envision this paradigm to be helpful in guiding the development of future novel HNE algorithms, and in the meantime facilitate their conceptual contrast towards existing ones.

As the second contribution, we prepare four benchmark heterogeneous network datasets through exhaustive data collection, cleaning, analysis and curation (Section 4). The datasets we come up with cover a wide spectrum of application domains (*i.e.*, publication, recommendation, knowledge base, and biomedicine), which have various properties regarding scale, structure, attribute/label availability, *etc.* This diverse set of data, together with a series of standard network mining tasks and evaluation metrics, constitute a handy and fair benchmark resource for future HNE algorithms.

As the third contribution, many existing HNE algorithms (including some very popular ones) either do not have a flexible implementation (*e.g.*, hard-coded node and edge types, fixed set of meta-paths, *etc.*), or do not scale to larger networks (*e.g.*, high memory requirement during training), which adds much burden to novel research (*i.e.*, requiring much engineering effort in correct reimplementation). To this end, we focus on eleven popular HNE algorithms, where we carefully refactor and scale up the original implementations and apply additional interfaces for plug-and-run experiments on our prepared datasets (Section 5). Based on these ready-to-use and efficient implementations, we then conduct all-around empirical evaluations of the algorithms, and report their benchmark performances. The empirical results, while providing much insight into the merits of different models that are consistent with the conceptual analysis in Section 3, also serve as the example utilization of our benchmark platform that can be followed by future studies on HNE.

Note that, although there have been several attempts to survey or benchmark heterogeneous network models [7], [8], [79], [87] and homogeneous graph embedding [26], [27], [28], [88], [89], none of them has deeply looked into the intersection of the two. We advocate that our unified framework for the research and experiments on HNE is timely and necessary. Firstly, as we will cover in this work, there has been a significant amount of research on the particular problem of HNE especially in the very recent several years, but most of them scatter across different domains, lacking proper connections and comparisons. Secondly, none of the existing surveys has proposed a generic mathematically complete paradigm for conceptual analysis of all HNE models. Thirdly, existing surveys mostly do not provide systematic benchmark evaluation results, nor do they come with benchmark datasets and open-source baselines to facilitate future algorithm development.

The rest of this paper is organized as follows. Section 2 first introduces our proposed generic HNE paradigm. Subsequently, representative models in our survey are conceptually categorized and analyzed in Section 3. We then present in Section 4 our prepared benchmark datasets with detailed analysis. In Section 5, we provide a systematic empirical study over eleven popular HNE algorithms to benchmark the current state-of-the-art of HNE. Section 6 concludes the paper with visions towards future usage of our platform and research on HNE.

2 Generic Paradigm

2.1 Problem Definitions

Definition 2.1. Heterogeneous network. A **heterogeneous network** $H = \{V, E, \phi, \psi\}$ is a network with multiple types of nodes and links. Particularly, within H, each node $v_i \in V$ is associated with a node type $\phi(v_i)$, and each link $e_{ij} \in E$ is associated with a link type $\psi(e_{ij})$. It is worth noting that the type of a link e_{ij} automatically defines the types of nodes v_i and v_j on its two ends.

Heterogeneous networks have been intensively studied due to its power of accommodating multi-modal multi-typed interconnected data. Besides the classic example of DBLP data used in most existing works as well as Figure 1, consider a different yet illustrative example from NYTimes in Figure 2.[2] Nodes in this heterogeneous network include

1. https://dblp.uni-trier.de/

2. https://www.nytimes.com/

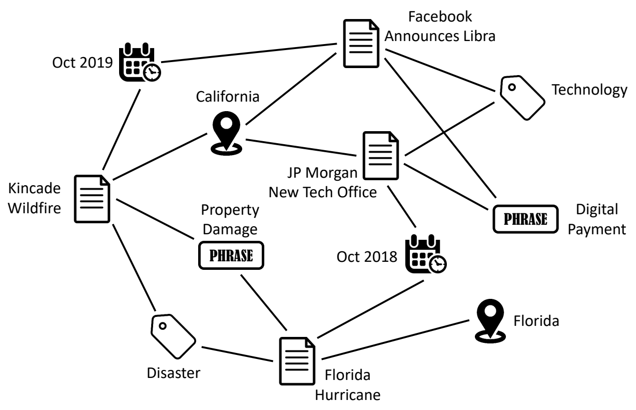


Fig. 2: **Toy example of a heterogeneous network constructed from the news data.**

news articles, categories, phrases, locations, and datetimes. To illustrate the power of heterogeneous networks, we introduce the concept of meta-path, which has been leveraged by most existing works on heterogeneous network modeling [7], [8].

Definition 2.2. Meta-path. A **meta-path** is a path defined on the network schema denoted in the form of $o_1 \xrightarrow{l_1} o_2 \xrightarrow{l_2} \cdots \xrightarrow{l_m} o_{m+1}$, where o and l are node types and link types, respectively.

Each meta-path captures the proximity among the nodes on its two ends from a particular semantic perspective. Continue with our example of heterogeneous network from news data in Figure 2. The meta-path of article $\xrightarrow{\text{belongs to}}$ category $\xrightarrow{\text{includes}}$ article carries different semantics from article $\xrightarrow{\text{mentions}}$ location $\xrightarrow{\text{mentioned by}}$ article. Thus, the leverage of different meta-paths allows heterogeneous network models to compute the multi-modal multi-typed node proximity and relation, which has been shown beneficial to many real-world network mining applications [15], [25], [90].

Next, we introduce the problem of general network embedding (representation learning).

Definition 2.3. Network embedding. For a given network $G = \{V, E\}$, where V is the set of nodes (vertices) and E is the set of links (edges), a **network embedding** is a mapping function $\Phi : V \mapsto \mathbb{R}^{|V|\times d}$, where $d \ll |V|$. This mapping Φ defines the latent representation (*a.k.a.* embedding) of each node $v \in V$, which captures network topological information in E.

In most cases, network proximity is the major topological information to be captured. For example, DeepWalk [29] captures the random-walk based node proximity and illustrates the 2-dim node representations learned on the famous Zachary's Karate network of small groups, where a clear correspondence between the node position in the input graph and learned embedding space can be observed. Various follow-up works have improved or extended DeepWalk, while a complete coverage of them is beyond the scope of this work. In this work, we focus on the embedding of heterogeneous networks.

Now we define the main problem of focus in this work, heterogeneous network embedding (HNE), which lies in the intersection between Def 2.1 and Def 2.3.

Definition 2.4. Heterogeneous network embedding. For a given heterogeneous network H, a **heterogeneous network embedding** is a set of mapping functions $\{\Phi_k : V_k \mapsto \mathbb{R}^{|V_k|\times d}\}_{k=1}^{K}$, where K is the number of node types, $\forall v_i \in V_k$, $\phi(v_i) = k$, $d \ll |V|$. Each mapping Φ_k defines the latent representation (*a.k.a.* embedding) of all nodes of type k, which captures the network topological information regarding the heterogeneous links in E.

Compared with homogeneous networks, the definition of topological information in heterogeneous networks is even more diverse. As we will show in Section 3, the major distinctions among different HNE algorithms mostly lie in their different ways of capturing such topological information. Particularly, the leverage of meta-paths as in Def 2.2 often plays an essential role, since many popular HNE algorithms exactly aim to model the different proximity indicated by meta-paths [59], [63], [66], [69], [70], [73], [75], [77].

2.2 Proposed Paradigm

In this work, we stress that one of the most important principles underlying HNE (as well as most other scenarios of network modeling and mining) is homophily [91]. Particularly, in the network embedding setting, homophily can be translated as *'nodes close on a network should have similar embeddings'*, which matches the requirement of Def 2.3. In fact, we further find intrinsic connections between the well-perceived homophily principle and widely-used smoothness enforcement technique on networks, which leads to a generic mathematical paradigm covering most existing and likely many future HNE algorithms.

Based on earlier well-established concepts underlying network modeling and embedding learning [92], [93], [94], [95], [96], we introduce the following key objective function of network smoothness enforcement as follows

$$\mathcal{J} = \sum_{u,v\in V} w_{uv} d(\boldsymbol{e}_u, \boldsymbol{e}_v) + \mathcal{J}_R, \tag{1}$$

where $\boldsymbol{e}_u = \Phi(u)$ and $\boldsymbol{e}_v = \Phi(v)$ are the node embedding vectors to be learned. w_{uv} is the proximity weight, $d(\cdot,\cdot)$ is the embedding distance function, and $\mathcal{J}_R$ denotes possible additional objectives such as regularizers, all three of which can be defined and implemented differently by the particular HNE algorithms.

3 Algorithm Taxonomy

In this section, we find a universal taxonomy for existing HNE algorithms with three categories based on their common objectives, and elaborate in detail how they all fit into our paradigm of Eq. (1). The main challenge of instantiating Eq. (1) on heterogeneous networks is the consideration of complex interactions regarding multi-typed links and higher-order meta-paths. In fact, our Eq. (1) also readily generalizes to homogeneous networks, though that is beyond the scope of this work.

3.1 Proximity-Preserving Methods

As mentioned above, one basic goal of network embedding is to capture network topological information. This can be achieved by preserving different types of proximity among nodes. There are two major categories of proximity-preserving methods in HNE: random walk approaches (inspired by DeepWalk [29]) and first/second-order proximity based ones (inspired by LINE [30]).

3.1.1 Random Walk Approaches

metapath2vec [59]. Following homogeneous network embedding [29], [31], metapath2vec utilizes the node paths traversed by meta-path guided random walks to model the context of a node regarding heterogeneous semantics. Formally, given a meta-path $\mathcal{M} = o_1 \xrightarrow{l_1} o_2 \xrightarrow{l_2} \cdots \xrightarrow{l_{m-1}} o_m$, the transition probability at step i is defined as

$$p(v_{i+1}|v_i, \mathcal{M}) = \begin{cases} \frac{1}{|\mathcal{N}_{l_i}(v_i)|} & \phi(v_{i+1}) = o_{i+1}, \psi(v_i, v_{i+1}) = l_i \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $\mathcal{N}_l(v) = \{u|\psi(u,v) = l\}$ denotes the neighbors of v associated with edge type l. Assume $\mathcal{P} = \{\mathcal{P}_1, ..., \mathcal{P}_M\}$ is the set of generated random walk sequences. The objective of metapath2vec is

$$\mathcal{J} = \sum_{v \in V} \sum_{u \in \mathcal{C}(v)} \log \frac{\exp(\boldsymbol{e}_u^T \boldsymbol{e}_v)}{\sum_{u' \in V} \exp(\boldsymbol{e}_{u'}^T \boldsymbol{e}_v)}, \tag{3}$$

where $\mathcal{C}(v)$ is the contexts (*i.e.*, skip-grams) of v in $\mathcal{P}$. For example, if $\mathcal{P}_1 = v_1 v_2 v_3 v_4 v_5 ...$ and the context window size is 2, then $\{v_1, v_2, v_4, v_5\} \subseteq \mathcal{C}(v_3)$. Let w_{uv} be the number of times that $u \in \mathcal{C}(v)$ is in $\mathcal{P}$. Eq. (3) can be rewritten as

$$\mathcal{J} = \sum_{u,v \in V} w_{uv} \log \frac{\exp(\boldsymbol{e}_u^T \boldsymbol{e}_v)}{\sum_{u' \in V} \exp(\boldsymbol{e}_{u'}^T \boldsymbol{e}_v)}.$$

The denominator in this objective requires summing over all nodes and is computationally expensive. In actual computation, it is approximated using negative sampling [97]. In this paper, we still analyze the original objective.

HIN2Vec [63]. HIN2Vec considers the probability that there is a meta-path $\mathcal{M}$ between nodes u and v. Specifically,

$$p(\mathcal{M}|u,v) = \sigma\Big(\mathbf{1}^T \Big(\boldsymbol{W}_X^T \boldsymbol{u} \odot \boldsymbol{W}_Y^T \boldsymbol{v} \odot f_{01}(\boldsymbol{W}_R^T \boldsymbol{m})\Big)\Big),$$

where $\mathbf{1}$ is an all-ones vector; $\odot$ is the Hadamard product; f_{01} is a normalization function. Here $\boldsymbol{e}_u = \boldsymbol{W}_X^T \boldsymbol{u}$, $\boldsymbol{e}_v = \boldsymbol{W}_Y^T \boldsymbol{v}$ and $\boldsymbol{e}_\mathcal{M} = f_{01}(\boldsymbol{W}_R^T \boldsymbol{m})$ can be viewed as the embeddings of u, v and $\mathcal{M}$, respectively. Let $\boldsymbol{A}_\mathcal{M} = diag(\boldsymbol{e}_{\mathcal{M}1}, ..., \boldsymbol{e}_{\mathcal{M}d})$. We have

$$p(\mathcal{M}|u,v) = \sigma\Big(\mathbf{1}^T (\boldsymbol{e}_u \odot \boldsymbol{e}_v \odot \boldsymbol{e}_\mathcal{M})\Big) = \sigma(\boldsymbol{e}_u^T \boldsymbol{A}_\mathcal{M} \boldsymbol{e}_v).$$

σ is the sigmoid function, so we have

$$1 - p(\mathcal{M}|u,v) = 1 - \sigma(\boldsymbol{e}_u^T \boldsymbol{A}_\mathcal{M} \boldsymbol{e}_v) = \sigma(-\boldsymbol{e}_u^T \boldsymbol{A}_\mathcal{M} \boldsymbol{e}_v).$$

HIN2Vec generates positive tuples $(u, v, \mathcal{M})$ (*i.e.*, u connects with v via meta-path $\mathcal{M}$) using homogeneous random walks [29] regardless of node/link types. For each positive tuple $(u, v, \mathcal{M})$, it generates several negative tuples by replacing u with a random node u'. Its objective is

$$\begin{aligned} \mathcal{J}_0 &= \sum_{(u,v,\mathcal{M})} \log p(\mathcal{M}|u,v) + \sum_{(u',v,\mathcal{M})} \log(1 - p(\mathcal{M}|u,v)) \\ &= \sum_{(u,v,\mathcal{M})} \Big(\log \sigma(\boldsymbol{e}_u^T \boldsymbol{A}_\mathcal{M} \boldsymbol{e}_v) + \sum_{u'} \log \sigma(-\boldsymbol{e}_{u'}^T \boldsymbol{A}_\mathcal{M} \boldsymbol{e}_v) \Big). \end{aligned}$$

This is actually the negative sampling approximation of the following objective

$$\mathcal{J} = \sum_{\mathcal{M}} \sum_{u,v \in V} w_{uv}^{\mathcal{M}} \log \frac{\exp(\boldsymbol{e}_u^T \boldsymbol{A}_\mathcal{M} \boldsymbol{e}_v)}{\sum_{u' \in V} \exp(\boldsymbol{e}_{u'}^T \boldsymbol{A}_\mathcal{M} \boldsymbol{e}_v)},$$

where $w_{uv}^{\mathcal{M}}$ is the number of path instances between u and v following meta-path $\mathcal{M}$.

Other random walk approaches are summarized in Table 1. To be specific, MRWNN [57] incorporates content priors into DeepWalk embedding; SHNE [69] incorporates additional node information like categorical attributes, images, *etc.* by leveraging domain-specific deep encoders; HHNE [73] extends metapath2vec to the hyperbolic space; GHE [19] proposes a semi-supervised meta-path weighting technique; MNE [98] conducts random walks separately for each view in a multi-view network; JUST [65] proposes Jump/Stay random walks that do not rely on pre-defined meta-paths; HeteSpaceyWalk [99] introduces a scalable embedding framework based on heterogeneous personalized spacey random walk; TapEm [100] proposes a task-guided node pair embedding approach for author identification.

3.1.2 First/Second-Order Proximity Based Approaches

PTE [101]. PTE proposes to decompose a heterogeneous network into multiple bipartite networks, each of which describes one edge type. Its objective is the sum of log-likelihoods over all bipartite networks:

$$\begin{aligned} \mathcal{J} &= \sum_{l \in \mathcal{T}_E} \sum_{u,v \in V} w_{uv}^l \log \frac{\exp(\boldsymbol{e}_u^T \boldsymbol{e}_v)}{\sum_{u' \in V_{\phi(u)}} \exp(\boldsymbol{e}_{u'}^T \boldsymbol{e}_v)} \\ &= \sum_{u,v \in V} w_{uv} \log \frac{\exp(\boldsymbol{e}_u^T \boldsymbol{e}_v)}{\sum_{u' \in V_{\phi(u)}} \exp(\boldsymbol{e}_{u'}^T \boldsymbol{e}_v)}. \end{aligned}$$

Here $\mathcal{T}_E$ is the set of edge types; w_{uv}^l is the type-l edge weight of (u, v) (if there is no edge between u and v with type l, then $w_{uv}^l = 0$); $w_{uv} = \sum_l w_{uv}^l$ is the total edge weight between u and v.

AspEm [60]. AspEm assumes that each heterogeneous network has multiple aspects, and each aspect is defined as a subgraph of the network schema [14]. An incompatibility measure is proposed to select appropriate aspects for embedding learning. Given an aspect a, its objective is

$$\mathcal{J} = \sum_{l \in \mathcal{T}_E^a} \frac{1}{Z_l} \sum_{u,v \in V} w_{uv}^l \log \frac{\exp(\boldsymbol{e}_{u,a}^T \boldsymbol{e}_{v,a})}{\sum_{u' \in V_{\phi(u)}} \exp(\boldsymbol{e}_{u,a}^T \boldsymbol{e}_{v,a})},$$

where $\mathcal{T}_E^a$ is the set of edge types in aspect a; $Z_l = \sum_{u,v} w_{uv}^l$ is a normalization factor; $e_{u,a}$ is the aspect-specific embedding of u.

Algorithm	w_{uv} / w_{uv}^{l} / $w_{uv}^{\mathcal{M}}$	$d(\boldsymbol{e}_u, \boldsymbol{e}_v)$	$\mathcal{J}_{R0}$	**Applications**
MRWNN [57]	number of times that $u \in \mathcal{C}(v)$ in homogeneous random walks	$\|\|\boldsymbol{e}_u - \boldsymbol{e}_v\|\|^2$	$-\sum_v \|\|\boldsymbol{e}_v - f_{\text{ENC}}(X_v)\|\|^2$	image retrieval
metapath2vec [59]	number of times that $u \in \mathcal{C}(v)$ in heterogeneous random walks following meta-path $\mathcal{M}$ (Eq. (2))		N/A	node classification, node clustering, link prediction, recommendation
SHNE [69]		$\|\|\boldsymbol{e}_u - \boldsymbol{e}_v\|\|^2$, $\boldsymbol{e}_u = f_{\text{ENC}}(\boldsymbol{x}_u)$		
HHNE [73]		$d_{\mathbb{D}}(\boldsymbol{e}_u, \boldsymbol{e}_v)$		
GHE [19]	number of meta-path instances between u and v following $\mathcal{M}$	$\|\|\boldsymbol{e}_u - \boldsymbol{e}_v\|\|^2$		author identification
HIN2Vec [63]		$\|\|\sqrt{\boldsymbol{A}_{\mathcal{M}}}(\boldsymbol{e}_u - \boldsymbol{e}_v)\|\|^2$		node classification, node clustering, link prediction, recommendation
MNE [98]	number of times that $u \in \mathcal{C}(v)$ in homogeneous random walks in (V, E_l)	$\|\|\boldsymbol{e}_u - \boldsymbol{e}_{v,l}\|\|^2$		
JUST [65]	number of times that $u \in \mathcal{C}(v)$ in Jump/Stay random walks [65]	$\|\|\boldsymbol{e}_u - \boldsymbol{e}_v\|\|^2$		
HeteSpaceyWalk [99]	number of times that $u \in \mathcal{C}(v)$ in heterogeneous spacey random walks [99]			
TapEm [100]	number of times that $u \in \mathcal{C}(v)$ in heterogeneous random walks following $\mathcal{M}$	$\|\|f_{\text{MAP}}(\boldsymbol{e}_u, \boldsymbol{e}_v) - \boldsymbol{e}_{u \to v}\|\|^2$ $\boldsymbol{e}_{u \to v}$: embedding of node sequence from u to v in $\mathcal{P}$	supervised loss $\sum y \log \hat{y}$ $+(1-y)\log(1-\hat{y})$	author identification
HNE [56]	$\mathbf{1}_{(u,v) \in E}$	$\|\|\boldsymbol{e}_u - \boldsymbol{e}_v\|\|^2$, $\boldsymbol{e}_u = \boldsymbol{A}_{\phi(u)} \boldsymbol{x}_u$	$-\sum_{o \in \mathcal{T}_V} \|\|\boldsymbol{A}_o\|\|_F^2$	text classification, image retrieval
PTE [101]	edge weight of (u, v)	$\|\|\boldsymbol{e}_u - \boldsymbol{e}_v\|\|^2$	N/A	text classification
DRL [68]				node classification
CMF [58]			$-\sum_v \|\|\boldsymbol{e}_v\|\|^2$	relatedness measurement of Wikipedia entities
HEBE [62]	edge weight of hyperedge (u, C) [62]	$\|\|\boldsymbol{e}_u - \boldsymbol{e}_C\|\|^2$, $\boldsymbol{e}_C = \sum_{v \in C} \boldsymbol{e}_v / \|C\|$	N/A	event embedding
Phine [11]	number of times that u and v co-occur in a meta-graph instance	$\|\|\boldsymbol{e}_u - \boldsymbol{e}_v\|\|^2$	supervised loss $-\sum \|\hat{y} - y\|^2$	user rating prediction
MVE [102]	edge weight of (u, v) with type l	$\|\|\boldsymbol{e}_{u,l} - \boldsymbol{e}_v\|\|^2$	$-\sum_v \sum_l \lambda_{v,l} \|\|\boldsymbol{e}_{v,l} - \boldsymbol{e}_v\|\|^2$	node classification, link prediction
AspEm [60]		$\|\|\boldsymbol{e}_{u,a} - \boldsymbol{e}_{v,a}\|\|^2$, a: aspect	N/A	aspect mining
HEER [61]		$\|\|\sqrt{\boldsymbol{A}_l}(\boldsymbol{e}_u - \boldsymbol{e}_v)\|\|^2$		relation prediction in knowledge graphs
mg2vec [103]	1st: number of meta-graph instances containing node u 2nd: number of meta-graph instances containing both nodes u and v	1st: $\|\|\boldsymbol{e}_v - \boldsymbol{e}_m\|\|^2$ 2nd: $\|\|f_{\text{MAP}}(\boldsymbol{e}_u, \boldsymbol{e}_v) - \boldsymbol{e}_m\|\|^2$ m: meta-graph		relationship prediction, relationship search

TABLE 1: A summary of proximity-preserving based HNE algorithms. (Additional notations: f_{ENC}: a function to encode text/image/attribute information; $d_{\mathbb{D}}$: distance between two points in the Poincaré ball; f_{MAP}: a function to map two d-dimensional embeddings to one $d-$dimenional vector.)

HEER [61]. HEER extends PTE by considering *typed closeness*. Specifically, each edge type l has an embedding $\boldsymbol{\mu}_l$, and its objective is

$$\mathcal{J} = \sum_{l \in \mathcal{T}_E} \sum_{u,v \in V} w_{uv}^{l} \log \frac{\exp(\boldsymbol{\mu}_l^T \boldsymbol{g}_{uv})}{\sum_{(u',v') \in P_l(u,v)} \exp(\boldsymbol{\mu}_l^T \boldsymbol{g}_{u'v'})},$$

where $\boldsymbol{g}_{uv}$ is the edge embedding of (u, v); $P_l(u, v) = \{(u', v) | \ \psi(u', v) = l\} \cup \{(u, v') | \psi(u, v') = l\}$. In [61], $\boldsymbol{g}_{uv}$ has different definitions for directed and undirected edges based on the Hadamard product. To simplify our discussion, we assume $\boldsymbol{g}_{uv} = \boldsymbol{e}_u \odot \boldsymbol{e}_v$. Let $\boldsymbol{A}_l = diag(\boldsymbol{\mu}_{l1}, ..., \boldsymbol{\mu}_{ld})$. We have $\boldsymbol{\mu}_l^T \boldsymbol{g}_{uv} = \boldsymbol{e}_u^T \boldsymbol{A}_l \boldsymbol{e}_v$, and

$$\mathcal{J} = \sum_{l \in \mathcal{T}_E} \sum_{u,v \in V} w_{uv}^{l} \log \frac{\exp(\boldsymbol{e}_u^T \boldsymbol{A}_l \boldsymbol{e}_v)}{\sum_{(u',v') \in P_l(u,v)} \exp(\boldsymbol{e}_{u'}^T \boldsymbol{A}_l \boldsymbol{e}_v)}.$$

There are many other first/second-order proximity based approaches summarized in Table 1. To be specific, Chang et al. [56] introduce a node type-aware content encoder; CMF [58] performs joint matrix factorization over the decomposed bipartite networks; HEBE [62] preserves proximity regarding each meta-graph; Phine [11] combines additional regularization towards semi-supervised training; MVE [102] proposes an attenion-based framework to consider multiple views of the same nodes; DRL [68] proposes to learn one type of edges at each step and uses deep reinforcement learning approaches to select the edge type for the next step; mg2vec [103] jointly embeds nodes and meta-graphs into the same space by exploiting both first- and second-order proximities.

3.1.3 Unified Objectives

Based on the above discussions, the objective of most proximity-preserving methods can be unified as

$$\max \mathcal{J} = \sum_{u,v} w_{uv} \log \frac{\exp(s(u, v))}{\sum_{u'} \exp(s(u', v))} + \mathcal{J}_{R0}. \quad (4)$$

Here, w_{uv} is the weight of node pair (u, v) in the objective[3]; $\mathcal{J}_{R0}$ is an algorithm-specific regularizer (summarized in Table 1); $s(u, v)$ is a proximity function between u and v.

Negative Sampling. Directly optimizing the objective in Eq. (4) is computationally expensive because it needs to traverse all nodes when computing the softmax function. Therefore, in actual computation, most studies adopt the negative sampling strategy [30], [97], which modify the objective as follows:

$$\sum_{u,v} w_{uv} \Big(\log \sigma(s(u,v)) + b\mathbb{E}_{u' \sim P_N}[\log \sigma(s(u',v))] \Big) + \mathcal{J}_{R0},$$

where b is the number of negative samples; P_N is known as the noise distribution that generates negative samples.

Negative sampling serves as a generic paradigm to unify network embedding approaches. For example, starting from the negative sampling objective, Qiu et al. [104] unify DeepWalk [29], LINE [30], PTE [101] and node2vec [31] into a matrix factorization framework. In this paper, as mentioned in Section 2.2, we introduce another objective (*i.e.*, Eq. (1)) that is equivalent to Eq. (4) and considers network embedding from the perspective of network smoothness enforcement.

Network Smoothness Enforcement. Again, we start from Eq. (4). Note that in most cases, we can write $s(u, v)$ as $f(\boldsymbol{e}_u)^T f(\boldsymbol{e}_v)$. For example, $f(\boldsymbol{e}_u) = \boldsymbol{e}_u$ in metapath2vec, PTE, *etc.*; $f(\boldsymbol{e}_u) = \sqrt{\boldsymbol{A}_\mathcal{M}}\boldsymbol{e}_u$ in HIN2Vec; $f(\boldsymbol{e}_u) = \sqrt{\boldsymbol{A}_l}\boldsymbol{e}_u$ in HEER. In these cases,

$$\begin{aligned}
\mathcal{J} &= \sum_{u,v} w_{uv} s(u,v) - \sum_{u,v} w_{uv} \log \sum_{u'} \exp(s(u',v)) + \mathcal{J}_{R0} \\
&= \sum_{u,v} w_{uv} f(\boldsymbol{e}_u)^T f(\boldsymbol{e}_v) \\
&\quad - \sum_{u,v} w_{uv} \log \sum_{u'} \exp(f(\boldsymbol{e}_{u'})^T f(\boldsymbol{e}_v)) + \mathcal{J}_{R0} \\
&= \sum_{u,v} \frac{w_{uv}}{2} \Big(||f(\boldsymbol{e}_u)||^2 + ||f(\boldsymbol{e}_v)||^2 - ||f(\boldsymbol{e}_u) - f(\boldsymbol{e}_v)||^2 \Big) \\
&\quad - \sum_{u,v} w_{uv} \log \sum_{u'} \exp(f(\boldsymbol{e}_{u'})^T f(\boldsymbol{e}_v)) + \mathcal{J}_{R0}.
\end{aligned}$$

The last step holds because $||\boldsymbol{x} - \boldsymbol{y}||^2 = (\boldsymbol{x} - \boldsymbol{y})^T(\boldsymbol{x} - \boldsymbol{y}) = ||\boldsymbol{x}||^2 + ||\boldsymbol{y}||^2 - 2\boldsymbol{x}^T\boldsymbol{y}$. Therefore, our goal is equivalent to

$$\begin{aligned}
\min -\mathcal{J} = &\sum_{u,v} \frac{w_{uv}}{2} \underbrace{||f(\boldsymbol{e}_u) - f(\boldsymbol{e}_v)||^2}_{d(\boldsymbol{e}_u, \boldsymbol{e}_v)} - \mathcal{J}_{R0} \\
&\underbrace{- \sum_{u,v} \frac{w_{uv}}{2} \Big(||f(\boldsymbol{e}_u)||^2 + ||f(\boldsymbol{e}_v)||^2 \Big)}_{\mathcal{J}_{R1}} \\
&\underbrace{+ \sum_{u,v} w_{uv} \log \sum_{u'} \exp(f(\boldsymbol{e}_{u'})^T f(\boldsymbol{e}_v))}_{\mathcal{J}_{R2}}.
\end{aligned} \tag{5}$$

Here $\mathcal{J}_{R1}$ and $\mathcal{J}_{R2}$ are two regularizers. Without $\mathcal{J}_{R1}$, $d(\boldsymbol{e}_u, \boldsymbol{e}_v)$ can be minimized by letting $||f(\boldsymbol{e}_v)|| \to 0$; without $\mathcal{J}_{R2}$, $d(\boldsymbol{e}_u, \boldsymbol{e}_v)$ can be minimized by letting $\boldsymbol{e}_v \equiv \boldsymbol{c}$ $(\forall v \in V)$. $\mathcal{J}_R$ in Eq. (1) is then the summation of the algorithm-specific regularizer $-\mathcal{J}_{R0}$ and $-\mathcal{J}_{R1}$, $\mathcal{J}_{R2}$, which are commonly shared across most HNE algorithms in the proximity-preserving group. We summarize the choices of w_{uv}, $d(e_u, e_v)$ and $\mathcal{J}_{R0}$ in Table 1.

Although Eq. (4) can cover most existing and likely many future approaches, we would like to remark that there are also studies adopting other forms of proximity-preserving objectives. For example, SHINE [105] uses reconstruction loss of autoencoders; HINSE [64] adopts spectral embedding based on adjacency matrices with different metagraphs; MetaDynaMix [106] introduces a low-rank matrix factorization framework for dynamic HIN embedding; HeGAN [72] proposes an adversarial learning approach with a relation type-aware discriminator.

3.2 Message-Passing Methods

Each node in a network can have attribute information represented as a feature vector $\boldsymbol{x}_u$. Message-passing methods aim to learn node embeddings $\boldsymbol{e}_u$ based on $\boldsymbol{x}_u$ by aggregating the information from u's neighbors. In recent studies, Graph Neural Networks (GNNs) [33] are widely adopted to facilitate this aggregation/message-passing process.

R-GCN [67]. R-GCN has K convolutional layers. The initial node representation $\boldsymbol{h}_u^{(0)}$ is just the node feature $\boldsymbol{x}_u$. In the k-th convolutional layer, each representation vector is updated by accumulating the vectors of neighboring nodes through a normalized sum.

$$\boldsymbol{h}_u^{(k+1)} = \sigma\Big(\sum_{l \in \mathcal{T}_E} \sum_{v \in \mathcal{N}_l(u)} \frac{1}{|\mathcal{N}_l(u)|} \boldsymbol{W}_l^{(k)} \boldsymbol{h}_v^{(k)} + \boldsymbol{W}_0^{(k)} \boldsymbol{h}_u^{(k)} \Big).$$

Different from regular GCNs [33], R-GCN considers edge heterogeneity by learning multiple convolution matrices $\boldsymbol{W}$'s, each corresponding to one edge type. During message passing, neighbors under the same edge type will be aggregated and normalized first. The node embedding is the output of the K-th layer (*i.e.*, $\boldsymbol{e}_v = \boldsymbol{h}_v^{(K)}$).

In unsupervised settings, message-passing approaches use link prediction as their downstream task to train GNNs. To be specific, the likelihood of observing edges in the heterogeneous network is maximized. R-GCN optimizes a cross-entropy loss through negative sampling. Essentially, it is the approximation of the following objective:

$$\mathcal{J} = \sum_{l \in \mathcal{T}_E} \sum_{u,v \in V} w_{uv}^l \log \frac{\exp(\boldsymbol{e}_u^T \boldsymbol{A}_l \boldsymbol{e}_v)}{\sum_{u' \in V} \exp(\boldsymbol{e}_u^T \boldsymbol{A}_l \boldsymbol{e}_v)},$$

where $w_{uv}^l = \mathbf{1}_{(u,v) \in E_l}$.

HAN [75]. Instead of considering one-hop neighbors, HAN utilizes meta-paths to model higher-order proximity. Given a meta-path $\mathcal{M}$, the representation of node u is aggregated from its meta-path based neighbors $\mathcal{N}_\mathcal{M}(u) = \{u\} \cup \{v | v$ connects with u via meta-path $\mathcal{M}\}$. HAN proposes an attention mechanism to learn the weights of different neighbors:

$$\alpha_{uv}^\mathcal{M} = \frac{\exp\big(\sigma(\boldsymbol{a}_\mathcal{M}^T [\boldsymbol{x}'_u || \boldsymbol{x}'_v])\big)}{\sum_{v' \in \mathcal{N}_\mathcal{M}(u)} \exp\big(\sigma(\boldsymbol{a}_\mathcal{M}^T [\boldsymbol{x}'_u || \boldsymbol{x}'_{v'}])\big)},$$
$$\boldsymbol{h}_u^\mathcal{M} = \sigma\Big(\sum_{v \in \mathcal{N}_\mathcal{M}(u)} \alpha_{uv}^\mathcal{M} \boldsymbol{x}'_v \Big),$$

where $\boldsymbol{a}_\mathcal{M}$ is the node-level attention vector of $\mathcal{M}$; $\boldsymbol{x}'_u = \boldsymbol{M}_{\phi(u)} \boldsymbol{x}_u$ is the projected feature vector of node u; $||$ is

3. w_{uv} can be specific to a meta-path $\mathcal{M}$ or an edge type l, in which cases we can denote it as $w_{uv}^\mathcal{M}$ or w_{uv}^l accordingly. In Eq. (4) and following derivations, for ease of notation, we omit the superscript.

Algorithm	w_{uv} / w_{uv}^l / $w_{uv}^{\mathcal{M}}$	$d(\boldsymbol{e}_u, \boldsymbol{e}_v)$	Aggregation Function	Applications
R-GCN [67]	$\mathbf{1}_{(u,v)\in E_l}$	$\lVert\sqrt{\boldsymbol{A}_l}(\boldsymbol{e}_u - \boldsymbol{e}_v)\rVert^2$	$\boldsymbol{h}_u^{(k+1)} = \sigma\Big(\sum_{l\in\mathcal{T}_E}\sum_{v\in\mathcal{N}_l(u)}\frac{1}{\lvert\mathcal{N}_l(u)\rvert}\boldsymbol{W}_r^{(k)}\boldsymbol{h}_v^{(k)} + \boldsymbol{W}_0^{(k)}\boldsymbol{h}_u^{(k)}\Big)$ $\boldsymbol{h}_u^{(0)} = \boldsymbol{x}_u,\ \ \boldsymbol{e}_u = \boldsymbol{h}_u^{(K)}$	entity classification, KB completion
HEP [107]	$\mathbf{1}_{(u,v)\in E}$	$\lVert\sqrt{\boldsymbol{A}}(\boldsymbol{e}_u - \boldsymbol{e}_v)\rVert^2$	$\boldsymbol{h}_u = \sigma\Big(\boldsymbol{W}_{\phi(u)}\Big(\Vert_{o\in\mathcal{T}_V}\sum_{v\in\mathcal{N}_o(u)}\alpha_{uv}\boldsymbol{e}_v\Big) + \boldsymbol{b}_{\phi(u)}\Big)$	user alignment
HAN [75]	edge weight of (u,v)	$\lVert\boldsymbol{e}_u - \boldsymbol{e}_v\rVert^2$	$\boldsymbol{e}_u = \sum_{\mathcal{M}}\beta_{\mathcal{M}}\sigma\Big(\sum_{v\in\mathcal{N}_{\mathcal{M}}(u)}\alpha_{uv}^{\mathcal{M}}\boldsymbol{M}_{\phi(u)}\boldsymbol{x}_u\Big)$	node classification, node clustering, link prediction, recommendation
HetGNN [71]	number of times that $u \in \mathcal{C}(v)$ in homogeneous random walks		$\boldsymbol{e}_u = \sum_{o\in\mathcal{T}_v}\alpha_u^o f_{\mathrm{AGG}}\Big(\{f_{\mathrm{ENC}}(\boldsymbol{x}_v) \vert v \in \mathcal{N}_{\mathrm{RWR}}(u), \phi(v) = o\}\Big)$	
GATNE [70]	number of times that $u \in \mathcal{C}(v)$ in random walks following $\mathcal{M}$	$\lVert\boldsymbol{e}_{u,l} - \boldsymbol{e}_v\rVert^2$	$\boldsymbol{h}_{u,l}^{(k+1)} = f_{\mathrm{AGG}}\Big(\{\boldsymbol{h}_{v,l}^{(k)} \vert v \in \mathcal{N}_l(u)\}\Big),\ \ \boldsymbol{h}_{u,l}^{(0)} = \boldsymbol{x}_u$ $\boldsymbol{e}_{u,l} = \alpha_l\boldsymbol{W}_l^T\Big(\Vert_{l\in\mathcal{T}_E}\boldsymbol{h}_{u,l}^{(K)}\Big)\boldsymbol{a}_{u,l} + \boldsymbol{b}_u$	
MAGNN [108]	edge weight of (u,v)	$\lVert\boldsymbol{e}_u - \boldsymbol{e}_v\rVert^2$	$\boldsymbol{h}_{u,\mathcal{M}} = \Vert_{k=1}^{K}\sigma\Big(\sum_{v\in\mathcal{N}_{\mathcal{M}}(u)}\alpha_{uv}^{\mathcal{M}} f_{\mathrm{ENC}}(\{\boldsymbol{M}_{\phi(t)}\boldsymbol{x}_t \vert t \in \mathcal{P}_{u\to v}^{\mathcal{M}}\})\Big)$ $\boldsymbol{e}_u = \sum_{\mathcal{M}}\beta_{\mathcal{M}}\sum_{o\in\mathcal{T}_V}\frac{1}{\lvert V_o\rvert}\boldsymbol{q}_o^T\tanh\Big(\boldsymbol{W}_o\boldsymbol{h}_{u,\mathcal{M}} + \boldsymbol{b}_o\Big)$	
HGT [85]	$\mathbf{1}_{(u,v)\in E}$	refer to NTN [20] in Table 3	$\hat{\boldsymbol{h}}_v^{(k+1)} = \sum_{u\in N(v)}\mathbf{Attention}(u,v)\odot\mathbf{Message}(u,v)$ $\boldsymbol{h}_v^{(k+1)} = A_{\phi(v)}(\sigma(\hat{\boldsymbol{h}}_v^{(k+1)})) + \boldsymbol{h}_v^{(k)},\ \ \boldsymbol{h}_u^{(0)} = \boldsymbol{x}_u,\ \ \boldsymbol{e}_u = \boldsymbol{h}_u^{(K)}$	node classification, author identification

TABLE 2: A summary of message-passing based HNE algorithms. (Additional notations: $\mathcal{N}_l(u)$: neighbors of u with edge type l; $\mathcal{N}_o(u)$: neighbors of u with node type o; $\mathcal{N}_{\mathcal{M}}(u)$: nodes connects with u via meta-path $\mathcal{M}$; $\mathcal{P}_{u\to v}^{\mathcal{M}}$: a meta-path instance connecting u and v; f_{AGG}: a function to aggregate information from neighbors. We show the transductive version of GATNE.)

the concatenation operator. Given meta-path specific embedding $\boldsymbol{h}_u^{\mathcal{M}}$, HAN uses a semantic-level attention to weigh different meta-paths:

$$\beta_{\mathcal{M}} = \frac{\exp\left(\frac{1}{|V|}\sum_{v\in V}\boldsymbol{q}^T\tanh(\boldsymbol{W}\boldsymbol{h}_v^{\mathcal{M}} + \boldsymbol{b})\right)}{\sum_{\mathcal{M}'}\exp\left(\frac{1}{|V|}\sum_{v\in V}\boldsymbol{q}^T\tanh(\boldsymbol{W}\boldsymbol{h}_v^{\mathcal{M}'} + \boldsymbol{b})\right)},$$
$$\boldsymbol{e}_u = \sum_{\mathcal{M}}\beta_{\mathcal{M}}\boldsymbol{h}_u^{\mathcal{M}},$$

where $\boldsymbol{q}$ is the semantic-level attention vector.

In the original HAN paper, the authors mainly consider the task of semi-supervised node classification. For unsupervised learning (*i.e.*, without any node labels), according to [108], HAN can use the link prediction loss introduced in GraphSAGE [34], which is approximation of the following objective:

$$\mathcal{J} = \sum_{u,v\in V} w_{uv}\log\frac{\exp(\boldsymbol{e}_u^T\boldsymbol{e}_v)}{\sum_{u'\in V}\exp(\boldsymbol{e}_{u'}^T\boldsymbol{e}_v)}. \qquad (6)$$

Here, w_{uv} is the edge weight of (u,v).

Recently, Ren et al. [86] propose HDGI to improve unsupervised training based on HAN by maximizing local-global mutual information.

HetGNN [71]. HetGNN assumes that each node is associated with different features (*e.g.*, categorical, textual and visual). It first encodes the content of each node to a vector $\boldsymbol{h}_u^s$, and then adopts a node type-aware aggregation function to collect information from neighbors:

$$\boldsymbol{h}_u^s = f_{\mathrm{ENC}}(\boldsymbol{x}_u),\ \ \boldsymbol{h}_u^o = f_{\mathrm{AGG}}(\{\boldsymbol{h}_v^s \vert v \in \mathcal{N}_{\mathrm{RWR}}(u), \phi(v) = o\}),$$

where $\mathcal{N}_{\mathrm{RWR}}(v)$ is the neighborhood of v defined through random walk with restart [109]. Then HetGNN uses attention over neighborhood node types to get the final embedding, which share the same spirits as HAN:

$$\alpha_u^o = \frac{\exp(\mathrm{LeakyReLU}(\boldsymbol{a}^T[\boldsymbol{h}_u^o \Vert \boldsymbol{h}_u^s]))}{\sum_{o'\in\mathcal{T}_V\cup\{s\}}\exp(\mathrm{LeakyReLU}(\boldsymbol{a}^T[\boldsymbol{h}_u^{o'} \Vert \boldsymbol{h}_u^s]))},$$
$$\boldsymbol{e}_u = \sum_{o\in\mathcal{T}_V\cup\{s\}}\alpha_u^o\boldsymbol{h}_u^o.$$

Following DeepWalk [29], HetGNN employs homogeneous random walk to generate node sequences, and its objective is the same as Eq. (6), except that w_{uv} means the number of times that $u \in \mathcal{C}(v)$ here.

HGT [85]. Inspired by the success of Transformer [110], [111] in text representation learning, Hu et al. propose to use each edge's type to parameterize the Transformer-like self-attention architecture. To be specific, for each edge (u,v), their Heterogeneous Graph Transformer (HGT) architecture maps v into a *Query* vector, and u into a *Key* vector, and calculate their dot product as attention:

$$\boldsymbol{Q}_v^i = Q_{\phi(v)}^i(\boldsymbol{h}_v^{(k)}), \quad \boldsymbol{K}_u^i = K_{\phi(u)}^i(\boldsymbol{h}_u^{(k)})$$
$$\mathrm{Head}_i^{ATT}(u,v) = \Big(\frac{\boldsymbol{K}_u^i\boldsymbol{W}_{\psi(u,v)}^{ATT}\boldsymbol{Q}_v^{i\ T}}{\sqrt{d}}\Big)\mu(\phi(u), \psi(u,v), \phi(v)),$$
$$\mathbf{Attention}(u,v) = \mathrm{Softmax}_{u\in N(v)}\Big(\Vert_i\mathrm{Head}_i^{ATT}(u,v)\Big),$$

where $\boldsymbol{h}_u^{(k)}$ is the output of the k-th HGT layer ($\boldsymbol{h}_u^{(0)} = \boldsymbol{x}_u$); $Q_{\phi(v)}^i$ and $K_{\phi(u)}^i$ are node type-aware linear mappings; Head_i^{ATT} is the i-th attention head; μ is a prior tensor representing the weight of each edge type in the attention. Parallel to the calculation of attention, the message passing process can be computed in a similar way by incorporating node and edge types:

$$\mathrm{Head}_i^{MSG}(u,v) = M_{\phi(u)}^i(\boldsymbol{h}_u^{(k)})\boldsymbol{W}_{\psi(u,v)}^{MSG},$$
$$\mathbf{Message}(u,v) = \Vert_i\mathrm{Head}_i^{MSG}(u,v),$$

where $M^i_{\phi(u)}$ is also a node type-aware linear mapping. To aggregate the messages from v's neighborhood, the attention vector serves as the weight to get the updated vector:

$$\hat{\boldsymbol{h}}_v^{(k+1)} = \sum_{u \in N(v)} \textbf{Attention}(u, v) \odot \textbf{Message}(u, v),$$

and following the residual connection [112], the output of the k-th layer is

$$\boldsymbol{h}_v^{(k+1)} = A_{\phi(v)}(\sigma(\hat{\boldsymbol{h}}_v^{(k+1)})) + \boldsymbol{h}_v^{(k)}.$$

Here $A_{\phi(u)}$ is a linear function mapping v's vector back to its node type-specific distribution.

For the unsupervised link prediction task, HGT borrows the objective function from Neural Tensor Network [20], which will be further discussed in Section 3.3.

There are some other message-passing approaches. For example, HEP [107] aggregates u's representation from $\mathcal{N}_o(u)$ (*i.e.*, the *node type-aware* neighborhood) with an application of e-commerce user alignment; GATNE [70] aggregates u's representation from $\mathcal{N}_l(u)$ (*i.e.*, the *edge type-aware* neighborhood) and is applicable for both transductive and inductive network embedding settings; MAGNN [108] aggregates u's representation from $\mathcal{N}_{\mathcal{M}}(u)$ (*i.e.*, the *meta-path-aware* neighborhood), and the nodes in between, by encoding the meta-path instances through a relational rotation encoder.

The objective of message-passing approaches mentioned above can also be written as Eq. (4) (except HGT, whose objective is the same as NTN [20] and will be discussed in Section 3.3), where $s(u, v)$ is a function of $\boldsymbol{e}_u$ and $\boldsymbol{e}_v$. The only difference is that $\boldsymbol{e}_u$ here is aggregated from $\boldsymbol{x}_v$ using GNNs. Following the derivation of proximity-preserving approaches, if we still write $\mathcal{J}_R$ in Eq. (1) as the summation of $-\mathcal{J}_{R_0}$, $-\mathcal{J}_{R_1}$ and $\mathcal{J}_{R_2}$, we can get the exactly same $\mathcal{J}_{R_1}$ and $\mathcal{J}_{R_2}$ as in Eq. (5).

Within this group of algorithms, only HEP has an additional reconstruction loss $\mathcal{J}_{R_0} = \sum_v ||\boldsymbol{e}_v - \boldsymbol{h}_v||^2$, while all other algorithms have $\mathcal{J}_{R_0} = 0$. We summarize the choices of w_{uv}, $d(\boldsymbol{e}_u, \boldsymbol{e}_v)$ and the aggregation function in Table 2.

3.3 Relation-Learning Methods

Each edge in a heterogeneous network can be viewed as a triplet (u, l, v) composed of two nodes $u, v \in V$ and an edge type $l \in \mathcal{T}_E$ (*i.e.*, entities and relations, in the terminology of KG). The goal of relation-learning methods is to learn a scoring function $s_l(u, v)$ which evaluates an arbitrary triplet and outputs a scalar to measure the acceptability of this triplet. This idea is widely adopted in KB embedding. Since there are surveys of KB embedding algorithms already [79], we only cover the most popular approaches here and highlight their connections to HNE.

TransE [4]. TransE assumes that the relation induced by l-labeled edges corresponds to a translation of the embedding (*i.e.*, $\boldsymbol{e}_u + \boldsymbol{e}_l \approx \boldsymbol{e}_v$) when (u, l, v) holds. Therefore, the scoring function of TransE is defined as

$$s_l(u, v) = -||\boldsymbol{e}_u + \boldsymbol{e}_l - \boldsymbol{e}_v||_p, \quad (7)$$

where $p = 1$ or 2. The objective is to minimize a margin-based ranking loss.

$$\mathcal{J} = \sum_{(u,l,v) \in T} \sum_{(u',l,v') \in T'_{(u,l,v)}} \max(0, \gamma - s_l(u, v) + s_l(u', v')), \quad (8)$$

where T is the set of positive triplets (*i.e.*, edges); $T'_{(u,l,v)}$ is the set of corrupted triplets, which are constructed by replacing either u or v with an arbitrary node. Formally,

$$T'_{(u,l,v)} = \{(u', l, v) | u' \in V\} \cup \{(u, l, v') | v' \in V\}.$$

TransE is the most representative model using "a translational distance" to define the scoring function. It has many extensions. For example, TransH [113] projects each entity vector to a relation-specific hyperplane when calculating the distance; TransR [80] further extends relation-specific hyperplanes to relation-specific spaces; RHINE [76] distinguishes affiliation relations from interaction relations and adopts different objectives for the two types of relations. For more extensions of TransE, please refer to [83].

Recently, Sun et al. [114] proposes RotatE model, which defines each relation as a rotation (instead of a translation) from the source entity to the target entity in the complex vector space. Their model is able to describe various relation patterns including symmetry/antisymmetry, inversion, and composition.

DistMult [81]. In contrast to translational distance models [4], [80], [113], DistMult exploits a similarity-based scoring function. Each relation is represented by a diagonal matrix $\boldsymbol{A}_l = diag(\boldsymbol{e}_{l1}, ..., \boldsymbol{e}_{ld})$, and $s_l(u, v)$ is defined using a bilinear function:

$$s_l(u, v) = \boldsymbol{e}_u^T \boldsymbol{A}_l \boldsymbol{e}_v.$$

Note that $s_l(u, v) = s_l(v, u)$ for any u and v. Therefore, DistMult is mainly designed for symmetric relations.

Besides DistMult, RESCAL [115] also uses a bilinear scoring function, but $\boldsymbol{A}_l$ is no longer restricted to be diagonal.

ConvE [82]. ConvE goes beyond simple distance or similarity functions and proposes deep neural models to score a triplet. The score is defined by a convolution over 2D shaped embeddings. Formally,

$$s_l(u, v) = f(\text{vec}(f([\boldsymbol{E}_u; \boldsymbol{E}_r] * \omega))\boldsymbol{W})\boldsymbol{e}_v,$$

where $\boldsymbol{E}_u$ and $\boldsymbol{E}_r$ denote the 2D reshaping matrices of node embedding and relation embedding, respectively; vec is the vectorization operator that maps a m by n matrix to a mn-dimensional vector; "$*$" is the convolution operator.

There are several other models leveraging deep neural scoring functions. For example, NTN [20] proposes to combine the two node embedding vectors by a relation-specific tensor $\underline{\boldsymbol{M}}_l \in \mathbb{R}^{d \times d \times d_l}$, where d_l is the dimension of $\boldsymbol{e}_l$; NKGE [83] develops a deep memory network to encode information from neighbors and employs a gating mechanism to integrate structure representation $\boldsymbol{e}_u^s$ and neighbor representation $\boldsymbol{e}_u^n$; SACN [84] proposes to encode node representations using a graph neural network and score a triplet using a convolutional network with the translational property.

Algorithm	w_{uv}^l	$d(\boldsymbol{e}_u, \boldsymbol{e}_v)$	$\mathcal{J}_{R0}$	**Applications**
TransE [4]	$\mathbf{1}_{(u,v)\in E_l}$	$\|\|\boldsymbol{e}_u + \boldsymbol{e}_l - \boldsymbol{e}_v\|\|$	$\sum_v(\|\|\boldsymbol{e}_v\|\| - 1)$	KB completion, relation extraction from text
TransH [113]		$\|\|\boldsymbol{e}_{u,l} + \boldsymbol{e}_l - \boldsymbol{e}_{v,l}\|\|^2$, $\boldsymbol{e}_{v,l} = \boldsymbol{e}_v - \boldsymbol{w}_l^T\boldsymbol{e}_v\boldsymbol{w}_l$	$\sum_l(\|\|\boldsymbol{w}_l\|\| - 1) + \sum_v[\|\|\boldsymbol{e}_v\|\| - 1]_+ + \sum_l\left[\frac{(\boldsymbol{w}_l^T\boldsymbol{e}_l)^2}{\|\|\boldsymbol{e}_l\|\|^2} - \epsilon^2\right]_+$	
TransR [80]		$\|\|\boldsymbol{e}_{u,l} + \boldsymbol{e}_l - \boldsymbol{e}_{v,l}\|\|^2$, $\boldsymbol{e}_{v,l} = \boldsymbol{A}_l\boldsymbol{e}_v$	$\sum_v[\|\|\boldsymbol{e}_v\|\| - 1]_+ + \sum_l[\|\|\boldsymbol{e}_l\|\| - 1]_+ + \sum_v[\|\|\boldsymbol{A}_l\boldsymbol{e}_v\|\| - 1]_+$	
RHINE [76]	edge weight of (u,v) with type l	$\|\|\boldsymbol{e}_u - \boldsymbol{e}_v\|\|^2$ if l models affiliation, $\|\|\boldsymbol{e}_u + \boldsymbol{e}_l - \boldsymbol{e}_v\|\|$ if l models interaction	N/A	link prediction, node classification
RotatE [114]	$\mathbf{1}_{(u,v)\in E_l}$	$\|\|\boldsymbol{e}_u \odot \boldsymbol{e}_l - \boldsymbol{e}_v\|\|^2$	$\sum_l(\|\|\boldsymbol{e}_l\|\| - 1)$	KB completion
RESCAL [115]		$\|\|\sqrt{\boldsymbol{A}_l}(\boldsymbol{e}_u - \boldsymbol{e}_v)\|\|^2$	$\sum_v[\|\|\boldsymbol{e}_v\|\| - 1]_+ + \sum_l[\|\|\boldsymbol{A}_l\|\|_F - 1]_+$	entity resolution, link prediction
DistMult [81]		$\|\|\sqrt{\boldsymbol{A}_l}(\boldsymbol{e}_u - \boldsymbol{e}_v)\|\|^2$, $\boldsymbol{A}_l = diag(\boldsymbol{e}_l)$	$\sum_v(\|\|\boldsymbol{e}_v\|\| - 1) + \sum_l[\|\|\boldsymbol{e}_l\|\| - 1]_+$	KB completion, triplet classification
NTN [20]		$C - \boldsymbol{e}_l^T\tanh\Big(\boldsymbol{e}_u^T\underline{\boldsymbol{M}}_l\boldsymbol{e}_v + \boldsymbol{M}_{l,1}\boldsymbol{e}_u + \boldsymbol{M}_{l,2}\boldsymbol{e}_v + \boldsymbol{b}_l\Big)$	$\|\|\Theta\|\|_2^2$	triplet classification
ConvE [82]		$C - \sigma\Big(\mathrm{vec}\big(\sigma([\boldsymbol{E}_u; \boldsymbol{E}_l] * \omega)\big)\boldsymbol{W}\Big)\boldsymbol{e}_v$	N/A	
NKGE [83]		same as TransE or ConvE, where $\boldsymbol{e}_u = \sigma(\boldsymbol{g}_u)\odot\boldsymbol{e}_u^s + \big(1 - \sigma(\boldsymbol{g}_u)\big)\odot\boldsymbol{e}_u^n$	$\|\|\Theta\|\|_2^2$	KB completion
SACN [84]		$C - f\Big(\mathrm{vec}\big(\boldsymbol{M}(\boldsymbol{e}_u, \boldsymbol{e}_l)\big)\boldsymbol{W}\Big)\boldsymbol{e}_v$	N/A	

TABLE 3: A summary of relation-learning based HNE algorithms. (Additional notations: f: a non-linear activation function; $[x]_+$: $\max\{x, 0\}$; $||\cdot||_F$: the Frobenius norm of a matrix; Θ: the set of learned parameters; $\boldsymbol{E}_u$, $\boldsymbol{E}_l$: 2D reshaping matrices of $\boldsymbol{e}_u$ and $\boldsymbol{e}_l$ [82]; vec: the vectorization operator; $*$: the convolution operator; $\boldsymbol{M}(\boldsymbol{e}_u, \boldsymbol{e}_l)$: a matrix aligning the output vectors from the convolution with all kernels [84].)

Most relation-learning based embedding algorithms adopt a margin-based ranking loss with some regularization terms that generalizes Eq. (8):

$$\sum_{(u,l,v)} w_{uv}^l \sum_{(u',l,v')} \max(0, \gamma - s_l(u,v) + s_l(u',v')) + \mathcal{J}_{R0}. \quad (9)$$

In [116], Qiu et al. point out that the margin-based loss shares a very similar form as the following negative sampling loss:

$$-\sum_{(u,l,v)} \Big(\log(\sigma(s_l(u,v))) - b\mathbb{E}_{(u',l,v')}\big[\log(\sigma(s_l(u',v')))\big]\Big).$$

Following [116], if we use use negative sampling loss to rewrite Eq. (9), we are approximately maximizing

$$\mathcal{J} = \sum_{(u,l,v)} w_{uv}^l \log \frac{\exp(s_l(u,v))}{\sum_{(u',l,v')}\exp(s_l(u',v'))} + \mathcal{J}_{R0}.$$

For translational distance models [4], [76], [80], [113] where $s_l(u,v)$ is described by distance, this is equivalent to

$$\min -\mathcal{J} = \sum_{(u,l,v)} w_{uv}^l \underbrace{||\boldsymbol{e}_u + \boldsymbol{e}_l - \boldsymbol{e}_v||^{1 \text{ or } 2}}_{d(\boldsymbol{e}_u, \boldsymbol{e}_v)} - \mathcal{J}_{R0}$$
$$+ \underbrace{\sum_{(u,l,v)} w_{uv}^l \log \sum_{(u',l,v')} \exp(s_l(u',v'))}_{\mathcal{J}_{R1}}. \quad (10)$$

In this case, we can write $\mathcal{J}_R$ as $-\mathcal{J}_{R0} + \mathcal{J}_{R1}$. For RotatE [114], the objective is the same except that $d(\boldsymbol{e}_u, \boldsymbol{e}_v) = ||\boldsymbol{e}_u \odot \boldsymbol{e}_l - \boldsymbol{e}_v||^2$.

For RESCAL [115] and DistMult [81], we can follow the derivation of Eq. (5) with $f(\boldsymbol{e}_u) = \sqrt{\boldsymbol{A}_l}\boldsymbol{e}_u$, and the objective will be

$$\min -\mathcal{J} = \sum_{(u,l,v)} \frac{w_{uv}^l}{2} \underbrace{||\sqrt{\boldsymbol{A}_l}(\boldsymbol{e}_u - \boldsymbol{e}_v)||^2}_{d(\boldsymbol{e}_u, \boldsymbol{e}_v)} - \mathcal{J}_{R0}$$
$$\underbrace{-\sum_{(u,l,v)} \frac{w_{uv}^l}{2}\Big(||f(\boldsymbol{e}_u)||^2 + ||f(\boldsymbol{e}_v)||^2\Big)}_{\mathcal{J}_{R1}}$$
$$\underbrace{+\sum_{(u,l,v)} w_{uv}^l \log \sum_{(u',l,v')} \exp(\boldsymbol{e}_{u'}^T\boldsymbol{A}_l\boldsymbol{e}_{v'})}_{\mathcal{J}_{R2}}.$$

The regularizer will be $\mathcal{J}_R = -\mathcal{J}_{R0} - \mathcal{J}_{R1} + \mathcal{J}_{R2}$.

For neural triplet scorers [20], [21], [82], [84], the forms of $s_l(u,v)$ are more complicated than translational distances or bilinear products. In these cases, since distance (or dissimilarity) and proximity can be viewed as reverse metrics, we define $d(\boldsymbol{e}_u, \boldsymbol{e}_v)$ as $C - s_l(u,v)$, where C is an constant upper bound of $s_l(\cdot,\cdot)$. Then the derivation of the loss function is similar to that of Eq. (10), i.e.,

$$\min -\mathcal{J} = \sum_{(u,l,v)} w_{uv}^l \underbrace{\Big(C - s_l(u,v)\Big)}_{d(\boldsymbol{e}_u, \boldsymbol{e}_v)} - \mathcal{J}_{R0}$$
$$\underbrace{+\sum_{(u,l,v)} w_{uv}^l \log \sum_{(u',l,v')} \exp(s_l(u',v'))}_{\mathcal{J}_{R1}}.$$

In this case, $\mathcal{J}_R = -\mathcal{J}_{R0} + \mathcal{J}_{R1}$.

We summarize the choices of w_{uv}^l, $d(\boldsymbol{e}_u, \boldsymbol{e}_v)$ and $\mathcal{J}_{R0}$ in Table 3. Note that for relation learning methods, $d(\cdot,\cdot)$ is usually *not* a distance metric. For example, $d(\boldsymbol{e}_u, \boldsymbol{e}_v) \neq d(\boldsymbol{e}_v, \boldsymbol{e}_u)$ in most translational distance models and deep neural models. This is intuitive because (u,l,v) and (v,l,u) often express different meanings.

4 Benchmark

4.1 Dataset Preparation

Towards off-the-shelf evaluation of HNE algorithms with standard settings, in this work, we collect, process, analyze, and publish four real-world heterogeneous network datasets from different domains, which we aim to set up as a handy and fair benchmark for existing and future HNE algorithms.

DBLP. We construct a network of authors, papers, venues, and phrases from DBLP. Phrases are extracted by the popular AutoPhrase [117] algorithm from paper texts and further filtered by human experts. We compute word2vec [97] on all paper texts and aggregate the word embeddings to get 300-dim paper and phrase features. Author and venue features are the aggregations of their corresponding paper features. We further manually label a relatively small portion of authors into 12 research groups from four research areas by crawling the web. Each labeled author has only one label.

Yelp. We construct a network of businesses, users, locations, and reviews from Yelp.[4] Nodes are not associated with any feature, but a large portion of businesses are labeled into sixteen categories. Each labeled business has one or multiple labels.

Freebase. We construct a network of books, films, music, sports, people, locations, organizations, and businesses from Freebase.[5] Nodes are not associated with any features, but a large portion of books are labeled into eight genres of literature. Each labeled book has only one label.

PubMed. We construct a network of genes, diseases, chemicals, and species from PubMed.[6] All nodes are extracted by AutoPhrase [117], typed by bioNER [118], and further filtered by human experts. The links are constructed through open relation pattern mining [119] and manual selection. We compute word2vec [97] on all PubMed papers and aggregate the word embeddings to get 200-dim features for all types of nodes. We further label a relatively small portion of diseases into eight categories. Each labeled disease has only one label.

The four datasets we prepare are from four different domains, which have been individually studied in some existing works on HNE. Among them, DBLP has been most commonly used, because all information about authors, papers, *etc.* is public and there is no privacy issue, and the results are often more interpretable to researchers in computer science related domains. Other real-world networks like Yelp and IMDB have been commonly studied for recommender systems. These networks are naturally heterogeneous, including at least users and items (*e.g.*, businesses and movies), as well as some additional item descriptors (*e.g.*, categories of businesses and genres of movies). Freebase is one of the most popular open-source knowledge graph, which is relatively smaller but cleaner compared with the others (*e.g.*, YAGO [120] and Wikidata [121]), where most entity and relation types are well defined. One major difference between conventional heterogeneous networks and knowledge graphs is the number of types of nodes and links. We further restrict the types of entities and relations inside Freebase, so as to get a heterogeneous network that is closer to a knowledge graph, while in the meantime does not have too many types of nodes and links. Therefore, most conventional HNE algorithms can be applied on this dataset and properly compared against the KB embedding ones. PubMed is a novel biomedical network we directly construct through text mining and manual processing on biomedical literature. This is the first time we make it available to the public, and we hope it to serve both the evaluation of HNE algorithms and novel downstream tasks in biomedical science such as biomedical information retrieval and disease evolution study.

4.2 Structure Analysis

A summary of the statistics on the four datasets is provided in Table 4 and Figure 3. As can be observed, the datasets have different sizes (numbers of nodes and links) and heterogeneity (numbers and distributions of node/link types). Moreover, due to the nature of the data sources, DBLP and PubMed networks are attributed, whereas Yelp and Freebase networks are abundantly labeled. A combination of these four datasets thus allows researchers to flexibly start by testing an HNE algorithm in the most appropriate settings, and eventually complete an all-around evaluation over all settings.

We also provide detailed analysis regarding several most widely concerned properties of heterogeneous networks, *i.e.*, degree distribution (Figure 4), clustering coefficient (Figure 5), and number of frequent meta-paths (Figure 6). In particular, degree distribution is known to significantly influence the performance of HNE algorithms due to the widely used node sampling process, whereas clustering coefficient impacts HNE algorithms that utilize latent community structures. Moreover, since many HNE algorithms rely on meta-paths, the skewer distribution of meta-paths can bias towards algorithms using fewer meta-paths.

As we can see, the properties we concern are rather different across the four datasets we prepare. For example, there are tighter links and more labels in Yelp, while there are more types of nodes and links in Freebase; compared with nodes in Freebase and PubMed which clearly follow the long-tail degree distribution, certain types of nodes in DBLP and Yelp are always well connected (*e.g.*, phrases in DBLP and businesses in Yelp), forming more *star-shaped* subgraphs; the type-wise clustering coefficients and meta-path distributions are the most skewed in DBLP and most balanced in PubMed. The set of four datasets together provide a comprehensive benchmark towards the robustness and generalizability of various HNE algorithms (as we will also see in Section 5.2).

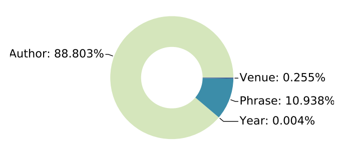
4.3 Settings, Tasks, and Metrics

We mainly compare all eleven algorithms under the setting of unsupervised unattributed HNE over all datasets, where the essential goal is to preserve different types of edges in the heterogeneous networks. Moreover, for message-passing algorithms that are particularly designed for attributed and semi-supervised HNE, we also conduct additional experiments for them in the corresponding settings.

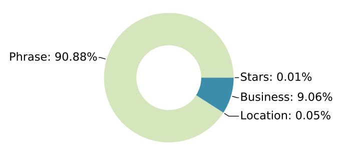
4. https://www.yelp.com/dataset/challenge
5. http://www.freebase.com/
6. https://www.ncbi.nlm.nih.gov/pubmed/

Dataset	#node type	#node	#link type	#link	#attributes	#attributed nodes	#label type	#labeled node
DBLP	4	1,989,077	6	275,940,913	300	ALL	13	618
Yelp	4	82,465	4	30,542,675	N/A	N/A	16	7,417
Freebase	8	12,164,758	36	62,982,566	N/A	N/A	8	47,190
PubMed	4	63,109	10	244,986	200	ALL	8	454

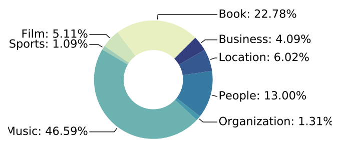
TABLE 4: **A summary of the statistics on four real-world heterogeneous network datasets.**



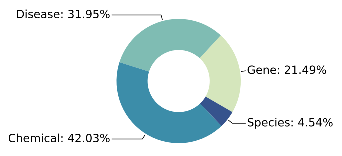
(a) DBLP



(b) Yelp

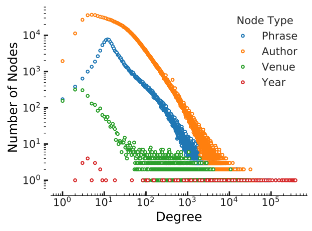


(c) Freebase

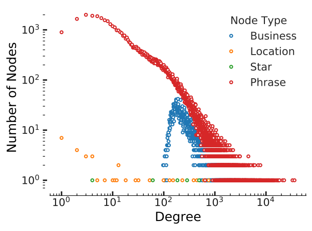


(d) PubMed

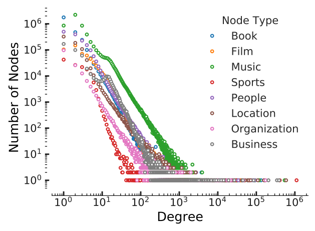
Fig. 3: **Portion of different node types in four real-world heterogeneous network datasets.**



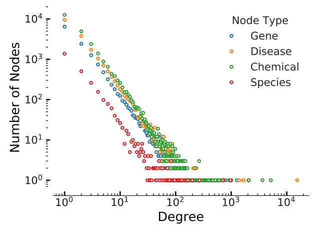
(a) DBLP



(b) Yelp

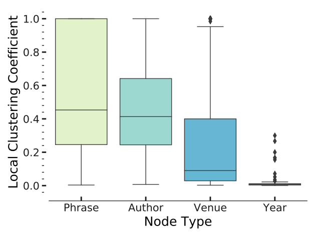


(c) Freebase

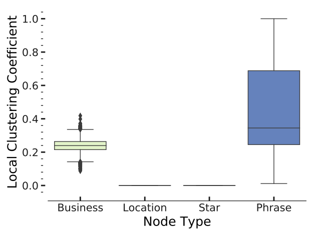


(d) PubMed

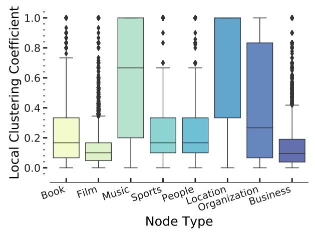
Fig. 4: **Degree distribution of different node types in four real-world heterogeneous network datasets.**



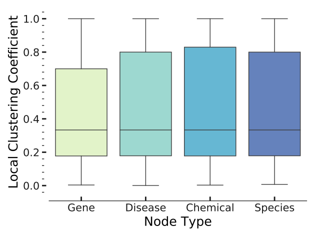
(a) DBLP



(b) Yelp

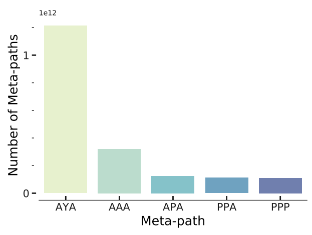


(c) Freebase

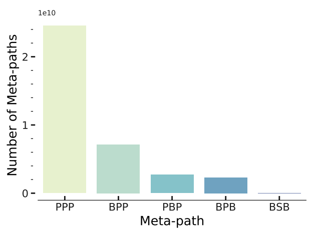


(d) PubMed

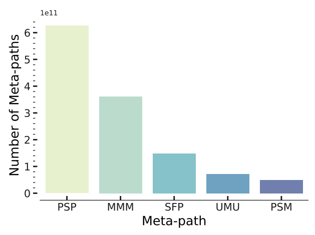
Fig. 5: **Local clustering coefficient of different node types in four real-world heterogeneous network datasets.**



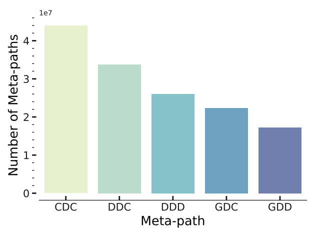
(a) DBLP



(b) Yelp



(c) Freebase



(d) PubMed

Fig. 6: **Number of five most frequent 2-hop meta-paths in four real-world heterogeneous network datasets.**

Particularly, due to the nature of the datasets, we evaluate attributed HNE on DBLP and PubMed datasets where node attributes are available, and semi-supervised HNE on Yelp and Freebase where node labels are abundant. We always test the computed network embeddings on the two standard network mining tasks of node classification and link prediction. We set the embedding size of all algorithms to 50 by default, and tune other hyperparameters following the original papers through standard five-fold cross validation on all datasets.

For the standard unattributed unsupervised HNE setting, we first randomly hide 20% links and train all HNE algorithms with the remaining 80% links. For node classification, we then train a separate linear Support Vector Machine (LinearSVC) [122] based on the learned embeddings on 80% of the labeled nodes and predict on the remaining 20%. We repeat the process for five times and compute the average scores regarding macro-F1 (across all labels) and micro-F1 (across all nodes). For link prediction, we use the Hadamard function to construct feature vectors for node pairs, train a two-class LinearSVC on the 80% training links and evaluate towards the 20% held out links. We also repeat the process for five times and compute the two metrics of AUC (area under the ROC curve) and MRR (mean reciprocal rank). AUC is a standard measure for classification, where we regard link prediction as a binary classification problem, and MRR is a standard measure for ranking, where we regard link prediction as a link retrieval problem. Since exhaustive computation over all node pairs is too heavy, we always use the two-hop neighbors as the candidates for all nodes.

For attributed HNE, node features are used during the training of HNE algorithms, while for semi-supervised HNE, certain amounts of node labels are used (80% by default).

5 Experimental Evaluations

5.1 Algorithms and Modifications

We amend the implementations of eleven popular HNE algorithms for seamless and efficient experimental evaluations on our prepared datasets. The algorithms we choose and the modifications we make are as follows.

- **metapath2vec** [59]: Since the original implementation contains a large amount of hard-coded data-specific settings such as node types and meta-paths, and the optimization is unstable and limited as it only examines one type of meta-path based context, we completely reimplement the algorithm. In particular, we first run random walks to learn the weights of different meta-paths based on the number of sampled instances, and then train the model using the unified loss function, which is a weighted sum over the loss functions of individual meta-paths. Both the random walk and meta-path-based embedding optimization are implemented with multi-threads in parallel.
- **PTE** [101]: Instead of accepting labeled texts as input and working on text networks with the specific three types of nodes (word, document, and label) and three types of links (word-word, document-word, and label-word), we revise the original implementation and allow the model to consume heterogeneous networks directly with arbitrary types of nodes and links.
- **HIN2Vec** [63]: We remove unnecessary data preprocessing codes and modify the original implementation so that the program first generates random walks, then trains the model, and finally outputs node embeddings only.
- **AspEm** [60]: We clean up the hard-coded data-specific settings in the original implementation and write a script to connect the different components of automatically selecting the aspects with the least incompatibilities, as well as learning, matching, and concatenating the embeddings based on different aspects.
- **HEER** [61]: We remove the hard-coded data-specific settings and largely simplify the data preprocessing step in the original implementation by skipping the knockout step and disentangling the graph building step.
- **R-GCN** [67]: The existing implementation from DGL [123] is only scalable to heterogeneous networks with thousands of nodes, due to the requirement of putting the whole graphs into memory during graph convolutions. To scale up R-GCN, we perform fixed-sized node and link sampling for batch-wise training following the framework of GraphSAGE [34].
- **HAN** [75]: Since the original implementation of HAN contains a large amount of hard-coded data-specific settings such as node types and meta-paths, and is unfeasible for large-scale datasets due to the same reason as R-GCN, we completely reimplement the HAN algorithm based on our implementation of R-GCN. In particular, we first automatically construct meta-path based adjacency lists for the chosen node type, and then sample the neighborhood for the seed nodes during batch-wise training.
- **HGT** [85]: The original PyG-based [124] implementation of HGT targets on the specific task of author disambiguation among the associated papers in a dynamic academic graph [125]. Therefore, we refactor it by removing hard-coded data-specific settings, assigning the same timestamp to all the nodes, and conducting training over all types of links.
- **TransE** [4]: We modify the OpenKE [126] implementation so that the model outputs node embeddings only.
- **DistMult** [81]: We remove the hard-coded data-specific settings and largely simplify the data preprocessing step in the original implementation.
- **ConvE** [82]: Same as for DistMult.

We have put the implementation of all compared algorithms in a python package and released them together with the datasets to constitute an open-source ready-to-use HNE benchmark.

5.2 Performance Benchmarks

We provide systematic experimental comparisons of the eleven popular state-of-the-art HNE algorithms across our four datasets, on the scenarios of unsupervised unattributed HNE, attributed HNE, and semi-supervised HNE.

Table 5 shows the performance of compared algorithms on unsupervised unattributed HNE, evaluated towards node classification and link prediction. We have the following observations.

From the perspective of compared algorithms:

Model	Node classification (Macro-F1/Micro-F1)				Link prediction (AUC/MRR)			
	DBLP	Yelp	Freebase	PubMed	DBLP	Yelp	Freebase	PubMed
metapath2vec	43.85/55.07	5.16/23.32	20.55/46.43	12.90/15.51	65.26/90.68	80.52/99.72	56.14/78.24	69.38/84.79
PTE	43.34/54.53	5.10/23.24	10.25/39.87	09.74/12.27	57.72/77.51	50.32/68.84	57.89/78.23	70.36/89.54
HIN2Vec	12.17/25.88	5.12/23.25	17.40/41.92	10.93/15.31	53.29/75.47	51.64/66.71	58.11/81.65	69.68/84.48
AspEm	33.07/43.85	5.40/23.82	23.26/45.42	11.19/14.44	67.20/91.46	76.10/95.18	55.80/77.70	68.31/87.43
HEER	09.72/27.72	5.03/22.92	12.96/37.51	11.73/15.29	53.00/72.76	73.72/95.92	55.78/78.31	69.06/88.42
R-GCN	09.38/13.39	5.10/23.24	06.89/38.02	10.75/12.73	50.50/73.35	72.17/97.46	50.18/74.01	63.33/81.19
HAN	07.91/16.98	5.10/23.24	06.90/38.01	09.54/12.18	50.24/73.10	N/A	51.50/74.13	65.85/85.33
HGT	15.17/32.05	5.07/23.12	23.06/46.51	11.24/18.72	59.98/83.13	79.00/99.66	55.68/79.46	73.00/88.05
TransE	22.76/37.18	5.05/23.03	31.83/52.04	11.40/15.16	63.53/86.29	69.13/83.66	52.84/75.80	67.95/84.69
DistMult	11.42/25.07	5.04/23.00	23.82/45.50	11.27/15.79	52.87/74.84	80.28/99.73	54.91/78.04	70.61/90.64
ConvE	12.42/26.42	5.09/23.02	24.57/47.61	13.00/14.49	54.03/75.31	78.55/99.70	54.29/76.11	71.81/89.82

TABLE 5: **Performance comparison (%) under the standard setting of unattributed unsupervised HNE.**

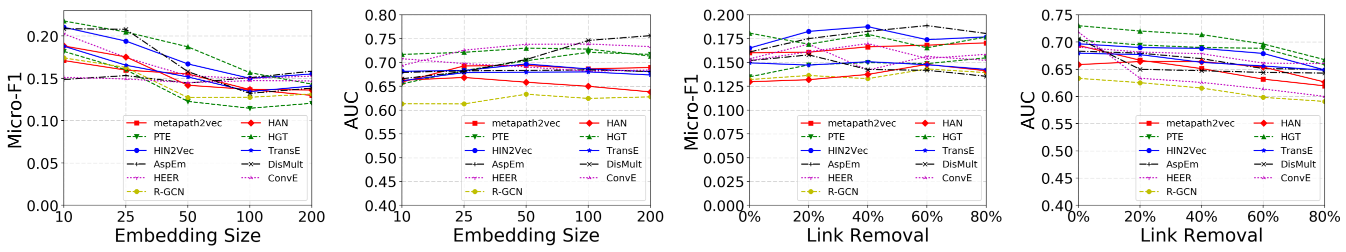


Fig. 7: **Performance comparison under controlled experiments with varying emb. sizes and link removals.**

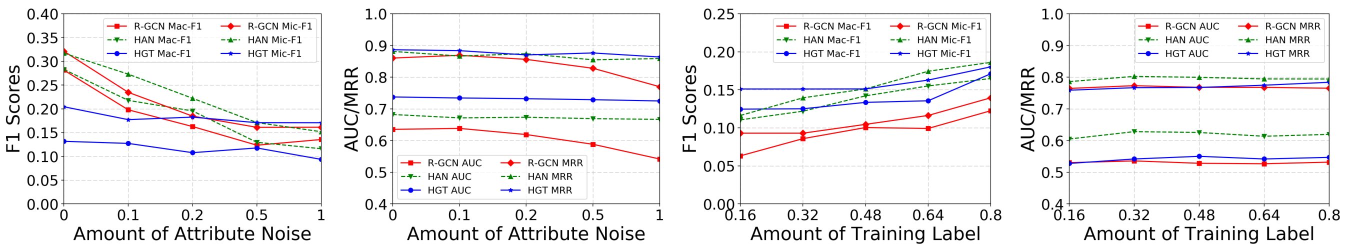


Fig. 8: **Performance comparison under controlled experiments with varying label amount and attr. noise.**

(1) Proximity-preserving algorithms often perform well on both tasks under the unsupervised unattributed HNE setting, which explains why proximity-preserving is the most widely used HNE or even general network embedding framework when node attributes and labels are unavailable. Among the proximity-preserving methods, HIN2Vec and HEER show reasonable results on link prediction but perform not so well on node classification (especially on DBLP and Freebase). In fact, these two methods focus on modeling link representations in their objectives ($\boldsymbol{A}_{\mathcal{M}}$ in HIN2Vec and $\boldsymbol{A}_l$ in HEER), thus are more suitable for link prediction.

(2) Under the unsupervised unattributed HNE setting, message-passing methods perform poorly except for HGT, especially on node classification. As we discuss before, message-passing methods are known to excel due to their integration of node attributes, link structures, and training labels. When neither of node attributes and labels are available, we use random vectors as node features and adopt a link prediction loss, which largely limits the performance of R-GCN and HAN. We will focus our evaluation on the message-passing algorithms in the attributed and semi-supervised HNE settings later. On the contrary, HGT exhibits competitive results on both node classification and link prediction. This is attributed to the usage of node-type and link-type dependent parameters which maintains dedicated representations for different types of nodes and links. In addition, the heterogeneous mini-batch graph sampling algorithm designed by HGT further reduces the loss of structural information due to sampling and boosts the performance to a greater extent. Finally, the link prediction result of HAN on the Yelp dataset is not available. This is because HAN can only embed one type of nodes at a time (we embed Business in Yelp) and thus predict the link between two nodes with the same type (*i.e.*, Business-Business). However, all links in Yelp connect distinct types of nodes (*e.g.*, Business-Location, Business-User), and HAN cannot predict such links.

(3) Relation-learning methods perform better on Freebase and PubMed on both tasks, especially on link prediction. In fact, in Table 4 and Figure 3 we can observe that both datasets (especially Freebase) have more link types. Relation-learning approaches, which are mainly designed to embed knowledge graphs (*e.g.*, Freebase), can better capture the semantics of numerous types of direct links.

From the perspective of datasets:

(1) All approaches have relatively low F1 scores on Yelp and Freebase (especially Yelp) on node classification. This is

because both datasets have larger numbers of labels (*i.e.*, 16 in Yelp and 8 in Freebase) as shown in Table 4. Moreover, unlike the cases of the other datasets, a node in Yelp can have multiple labels, which makes the classification task more challenging.

(2) In Figure 4, we can observe that the degree distribution of Freebase is more skewed. Therefore, when we conduct link sampling or random walks on Freebase during representation learning, nodes with lower degrees will be sampled less frequently and their representations may not be learned accurately. This observation may explain why the link prediction metrics on Freebase are in general lower than those on DBLP and Yelp.

(3) As we can see in Figure 3-6, most studied network properties are more balanced on Freebase and PubMed (especially PubMed) across different types of nodes and links. This in general makes both the node classification and link prediction tasks harder for all algorithms, and also makes the gaps among different algorithms smaller.

5.3 Ablation Studies

To provide an in-depth performance comparison among various HNE algorithms, we further conduct controlled experiments by varying the embedding sizes and randomly removing links from the training set.

In Figure 7, we show the micro-F1 scores for node classification and AUC scores for link prediction computed on the PubMed dataset. We omit the other results here, which can be easily computed in our provided benchmark package. As we can observe, some algorithms are more robust to varying settings while some others are more sensitive. In general, varying embedding size and link removal can significantly impact the performance of most algorithms on both tasks, and sometimes can even lead to different ordering of certain algorithms. This again emphasizes the importance of setting up standard benchmark including datasets and evaluation protocols for systematic HNE algorithm evaluation. In particular, on PubMed, larger embedding sizes like over 50 can harm the performance of most algorithms especially on node classification, probably due to overfitting with the limited labeled data. Interestingly, the random removal of links does have a negative impact on link prediction, but it does not necessarily harm node classification. This means that node classes and link structures may not always be tightly correlated, and even parts of the links already provide the necessary information useful enough for node classification.

Towards the evaluation of recent HNE algorithms that integrate node attributes and labels into representation learning like R-GCN, HAN, and HGT, we also conduct controlled experiments by adding random Gaussian noises to the node attributes and masking different amounts of training labels.

In Figure 8, we show the results on the PubMed dataset. As we can observe, the scores in most subfigures are significantly higher than the scores in Table 5, indicating the effectiveness of R-GCN, HAN, and HGT in integrating node attributes and labels for HNE. In particular, the incorporation of node attributes boosts the node classification results of R-GCN and HAN significantly but offers very little help to HGT. This suggests that R-GCN and HAN can better leverage the semantic information associated with attributes, whereas HGT relies more on the network structures and type information of nodes and links. Furthermore, when random noises with larger variances are added to node attributes, the performance of node classification significantly drops, while the performance of link prediction is less affected. As more training labels become available, without a surprise, the node classification results of all three algorithms increase, but surprisingly the link prediction results are almost not affected. These observations again reveal the different natures of the two tasks, where node classes are more related to node contents, whereas links should be typically inferred from structural information.

6 Future

In this work, we present a comprehensive survey on various existing HNE algorithms, and provide benchmark datasets and baseline implementations to ease future research in this direction. While HNE has already demonstrated strong performance across a variety of downstream tasks, it is still in its infancy with many open challenges. To conclude this work and inspire future research, we now briefly discuss the limitation of current HNE and several specific directions potentially worth pursuing.

Beyond homophily. As we formulate in Eq. (1), current HNE algorithms focus on the leverage of network homophily. Due to recent research on homogeneous networks that study the combination of *positional* and *structural* embedding, it would be interesting to explore how to generalize such design principles and paradigms to HNE. Particularly, in heterogeneous networks, relative positions and structural roles of nodes can both be measured under different meta-paths or meta-graphs, which are naturally more informative and diverse. However, such considerations also introduce harder computational challenges.

Beyond accuracy. Most, if not all, existing research on HNE has primarily focused on the accuracy towards different downstream tasks. It would be interesting to further study the *efficiency and scalability* (for large-scale networks), *temporal adaptability* (for dynamic evolving networks), *robustness* (towards adversarial attacks), *explainability*, *uncertainty*, *fairness* of HNE, and so on.

Beyond node embedding. Graph- and subgraph-level embeddings have been intensively studied on homogeneous networks, but hardly on heterogeneous networks. Although existing works like HIN2Vec [63] study the embedding of meta-paths to improve the embedding of nodes, direct applications of graph- and subgraph-level embeddings in the context of heterogeneous networks remain nascent.

Revisiting KB embedding. The difference between KB embedding and other types of HNE is mainly due to the numbers of node and link types. Direct application of KB embedding to heterogeneous networks fails to consider meta-paths with rich semantics, whereas directly applying HNE to KB is unrealistic due to the exponential number of meta-paths. However, it would still be interesting to study the intersection between these two groups of methods (as well as two types of data). For example, how can we combine the ideas of meta-paths on heterogeneous networks and embedding transformation on KB for HNE with more semantic-aware transformations? How can we devise truncated random walk based methods for KB embedding to include higher-order relations?

Modeling heterogeneous contexts. Heterogeneous networks mainly model different types of nodes and links. However, networks nowadays are often associated with rich contents, which provide contexts of the nodes, links, and subnetworks. Thus, how to model heterogeneous interactions under multi-facet contexts through the integration of multi-modal content and structure could be a challenging but rewarding research area.

Understanding the limitation. While HNE (as well as many neural representation learning models) has demonstrated strong performance in various domains, it is worthwhile to understand its potential limits. For example, when do modern HNE algorithms work better compared with traditional network mining approaches (*e.g.*, path counting, subgraph matching, non-neural or linear propagation)? How can we join the advantages of both worlds? Moreover, while there has been intensive research on the mathematical mechanisms behind neural networks for homogeneous network data (*e.g.*, smoothing, low-pass filtering, invariant and equivariant transformations), by unifying existing models on HNE, this work also aims to stimulate further theoretical studies on the power and limitation of HNE.

Acknowledgement

Research was sponsored in part by the U.S. Army Research Laboratory under Cooperative Agreement No. W911NF-09-2-0053 and W911NF-13-1-0193, DARPA No. W911NF-17-C-0099 and FA8750-19-2-1004, National Science Foundation IIS 16-18481, IIS 17-04532, and IIS-17-41317, DTRA HDTRA11810026, and grant 1U54GM114838 awarded by NIGMS through funds provided by the trans-NIH Big Data to Knowledge (BD2K) initiative. Special thanks to Dr Hanghang Tong for his generous help in the polish of this work.

References

[1] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson, "Structured sequence modeling with graph convolutional recurrent networks," in *ICONIP*, 2018.
[2] J. He, M. Li, H.-J. Zhang, H. Tong, and C. Zhang, "Manifold-ranking based image retrieval," in *SIGMM*, 2004.
[3] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *ICML*, 2017.
[4] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *NIPS*, 2013.
[5] C. Yang, L. Bai, C. Zhang, Q. Yuan, and J. Han, "Bridging collaborative filtering and semi-supervised learning: A neural approach for poi recommendation," in *KDD*, 2017.
[6] Y. Xiao, A. Krishnan, and H. Sundaram, "Discovering strategic behaviors for collaborative content-production in social networks," in *WWW*, 2020.
[7] Y. Sun and J. Han, "Mining heterogeneous information networks: principles and methodologies," *Synthesis Lectures on Data Mining and Knowledge Discovery*, vol. 3, no. 2, pp. 1–159, 2012.
[8] C. Shi, Y. Li, J. Zhang, Y. Sun, and S. Y. Philip, "A survey of heterogeneous information network analysis," *TKDE*, vol. 29, no. 1, pp. 17–37, 2016.
[9] J.-D. Zhang and C.-Y. Chow, "Geosoca: Exploiting geographical, social and categorical correlations for point-of-interest recommendations," in *SIGIR*, 2015.
[10] C. Yang, D. H. Hoang, T. Mikolov, and J. Han, "Place deduplication with embeddings," in *WWW*, 2019.
[11] C. Yang, C. Zhang, X. Chen, J. Ye, and J. Han, "Did you enjoy the ride: Understanding passenger experience via heterogeneous network embedding," in *ICDE*, 2018.
[12] Y. Li and J. C. Patra, "Genome-wide inferring gene–phenotype relationship by walking on the heterogeneous network," *Bioinformatics*, vol. 26, no. 9, pp. 1219–1224, 2010.
[13] A. P. Davis, C. J. Grondin, R. J. Johnson, D. Sciaky, B. L. King, R. McMorran, J. Wiegers, T. C. Wiegers, and C. J. Mattingly, "The comparative toxicogenomics database: update 2017," *Nucleic acids research*, vol. 45, no. D1, pp. D972–D978, 2016.
[14] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu, "Pathsim: Meta path-based top-k similarity search in heterogeneous information networks," in *VLDB*, 2011.
[15] C. Shi, X. Kong, Y. Huang, S. Y. Philip, and B. Wu, "Hetesim: A general framework for relevance measure in heterogeneous networks," *TKDE*, vol. 26, no. 10, pp. 2479–2492, 2014.
[16] C. Yang, M. Liu, F. He, X. Zhang, J. Peng, and J. Han, "Similarity modeling on heterogeneous networks via automatic path discovery," in *ECML-PKDD*, 2018.
[17] L. Dos Santos, B. Piwowarski, and P. Gallinari, "Multilabel classification on heterogeneous graphs with gaussian embeddings," in *ECML-PKDD*, 2016.
[18] D. Eswaran, S. Günnemann, C. Faloutsos, D. Makhija, and M. Kumar, "Zoobp: Belief propagation for heterogeneous networks," in *VLDB*, 2017.
[19] T. Chen and Y. Sun, "Task-guided and path-augmented heterogeneous network embedding for author identification," in *WSDM*, 2017.
[20] R. Socher, D. Chen, C. D. Manning, and A. Ng, "Reasoning with neural tensor networks for knowledge base completion," in *NIPS*, 2013.
[21] B. Oh, S. Seo, and K.-H. Lee, "Knowledge graph completion by context-aware convolutional learning with multi-hop neighborhoods," in *CIKM*, 2018.
[22] W. Zhang, B. Paudel, L. Wang, J. Chen, H. Zhu, W. Zhang, A. Bernstein, and H. Chen, "Iteratively learning embeddings and rules for knowledge graph reasoning," in *WWW*, 2019.
[23] X. Geng, H. Zhang, J. Bian, and T.-S. Chua, "Learning image and user features for recommendation in social networks," in *ICCV*, 2015.
[24] H. Zhao, Q. Yao, J. Li, Y. Song, and D. L. Lee, "Meta-graph based recommendation fusion over heterogeneous information networks," in *KDD*, 2017.
[25] S. Hou, Y. Ye, Y. Song, and M. Abdulhayoglu, "Hindroid: An intelligent android malware detection system based on structured heterogeneous information network," in *KDD*, 2017.
[26] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *Knowledge-Based Systems*, vol. 151, pp. 78–94, 2018.
[27] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *TKDE*, vol. 30, no. 9, pp. 1616–1637, 2018.
[28] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *TKDE*, vol. 31, no. 5, pp. 833–852, 2018.
[29] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *KDD*, 2014.
[30] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *WWW*, 2015.
[31] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *KDD*, 2016.
[32] H. Wang, J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, X. Xie, and M. Guo, "Graphgan: Graph representation learning with generative adversarial nets," in *AAAI*, 2018.
[33] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.
[34] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NIPS*, 2017.
[35] J. Chen, T. Ma, and C. Xiao, "Fastgcn: fast learning with graph convolutional networks via importance sampling," in *ICLR*, 2018.
[36] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," in *ICLR*, 2019.

[37] N. Zhao, H. Zhang, M. Wang, R. Hong, and T.-S. Chua, "Learning content–social influential features for influence analysis," *IJMIR*, vol. 5, no. 3, pp. 137–149, 2016.

[38] S. Abu-El-Haija, B. Perozzi, and R. Al-Rfou, "Learning edge representations via low-rank asymmetric projections," in *CIKM*, 2017.

[39] B. Perozzi, V. Kulkarni, H. Chen, and S. Skiena, "Don't walk, skip!: online learning of multi-scale network embeddings," in *ASONAM*, 2017.

[40] C. Yang, J. Zhang, H. Wang, S. Li, M. Kim, M. Walker, Y. Xiao, and J. Han, "Relation learning on social networks with multi-modal graph edge variational autoencoders," in *WSDM*, 2020.

[41] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *ICML*, 2016.

[42] C. Yang, M. Liu, V. W. Zheng, and J. Han, "Node, motif and subgraph: Leveraging network functional blocks through structural convolution," in *ASONAM*, 2018.

[43] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *NIPS*, 2018.

[44] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe, "Weisfeiler and leman go neural: Higher-order graph neural networks," in *AAAI*, 2019.

[45] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in *CIKM*, 2015.

[46] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *KDD*, 2016.

[47] Z. Zhang, P. Cui, X. Wang, J. Pei, X. Yao, and W. Zhu, "Arbitrary-order proximity preserved network embedding," in *KDD*, 2018.

[48] J. Huang, X. Liu, and Y. Song, "Hyper-path-based representation learning for hyper-networks," in *CIKM*, 2019.

[49] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo, "struc2vec: Learning node representations from structural identity," in *KDD*, 2017.

[50] M. Zhang and Y. Chen, "Weisfeiler-lehman neural machine for link prediction," in *KDD*, 2017.

[51] T. Lyu, Y. Zhang, and Y. Zhang, "Enhancing the network embedding quality with structural similarity," in *CIKM*, 2017.

[52] C. Donnat, M. Zitnik, D. Hallac, and J. Leskovec, "Learning structural node embeddings via diffusion wavelets," in *KDD*, 2018.

[53] B. Scholkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

[54] A. Liaw, M. Wiener *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[55] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *KDD*, 2016.

[56] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, and T. S. Huang, "Heterogeneous network embedding via deep architectures," in *KDD*, 2015.

[57] F. Wu, X. Lu, J. Song, S. Yan, Z. M. Zhang, Y. Rui, and Y. Zhuang, "Learning of multimodal representations with random walks on the click graph," *TIP*, vol. 25, no. 2, pp. 630–642, 2015.

[58] Y. Zhao, Z. Liu, and M. Sun, "Representation learning for measuring entity relatedness with rich information," in *AAAI*, 2015.

[59] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *KDD*, 2017.

[60] Y. Shi, H. Gui, Q. Zhu, L. Kaplan, and J. Han, "Aspem: Embedding learning by aspects in heterogeneous information networks," in *SDM*, 2018.

[61] Y. Shi, Q. Zhu, F. Guo, C. Zhang, and J. Han, "Easing embedding learning by comprehensive transcription of heterogeneous information networks," in *KDD*, 2018.

[62] H. Gui, J. Liu, F. Tao, M. Jiang, B. Norick, and J. Han, "Large-scale embedding learning in heterogeneous event data," in *ICDM*, 2016.

[63] T.-y. Fu, W.-C. Lee, and Z. Lei, "Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning," in *CIKM*, 2017.

[64] C. Yang, Y. Feng, P. Li, Y. Shi, and J. Han, "Meta-graph based hin spectral embedding: Methods, analyses, and insights," in *ICDM*, 2018.

[65] R. Hussein, D. Yang, and P. Cudré-Mauroux, "Are meta-paths necessary? revisiting heterogeneous graph embeddings," in *CIKM*, 2018.

[66] Y. Zhang, Y. Xiong, X. Kong, S. Li, J. Mi, and Y. Zhu, "Deep collective classification in heterogeneous information networks," in *WWW*, 2018.

[67] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *ESWC*, 2018.

[68] M. Qu, J. Tang, and J. Han, "Curriculum learning for heterogeneous star network embedding via deep reinforcement learning," in *WSDM*, 2018.

[69] C. Zhang, A. Swami, and N. V. Chawla, "Shne: Representation learning for semantic-associated heterogeneous networks," in *WSDM*, 2019.

[70] Y. Cen, X. Zou, J. Zhang, H. Yang, J. Zhou, and J. Tang, "Representation learning for attributed multiplex heterogeneous network," in *KDD*, 2019.

[71] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla, "Heterogeneous graph neural network," in *KDD*, 2019.

[72] B. Hu, Y. Fang, and C. Shi, "Adversarial learning on heterogeneous information networks," in *KDD*, 2019.

[73] X. Wang, Y. Zhang, and C. Shi, "Hyperbolic heterogeneous information network embedding," in *AAAI*, 2019.

[74] C. Yang, J. Zhang, and J. Han, "Neural embedding propagation on heterogeneous networks," in *ICDM*, 2019.

[75] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous graph attention network," in *WWW*, 2019.

[76] Y. Lu, C. Shi, L. Hu, and Z. Liu, "Relation structure-aware heterogeneous information network embedding," in *AAAI*, 2019.

[77] C. Shi, B. Hu, W. X. Zhao, and S. Y. Philip, "Heterogeneous information network embedding for recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 2, pp. 357–370, 2018.

[78] Y. Cao, X. Wang, X. He, Z. Hu, and T.-S. Chua, "Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences," in *WWW*, 2019.

[79] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *TKDE*, vol. 29, no. 12, pp. 2724–2743, 2017.

[80] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *AAAI*, 2015.

[81] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," in *ICLR*, 2015.

[82] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2d knowledge graph embeddings," in *AAAI*, 2018.

[83] K. Wang, Y. Liu, X. Xu, and D. Lin, "Knowledge graph embedding with entity neighbors and deep memory network," in *AAAI*, 2019.

[84] C. Shang, Y. Tang, J. Huang, J. Bi, X. He, and B. Zhou, "End-to-end structure-aware convolutional networks for knowledge base completion," in *AAAI*, 2019.

[85] Z. Hu, Y. Dong, K. Wang, and Y. Sun, "Heterogeneous graph transformer," in *WWW*, 2020.

[86] Y. Ren, B. Liu, C. Huang, P. Dai, L. Bo, and J. Zhang, "Heterogeneous deep graph infomax," *arXiv preprint arXiv:1911.08538*, 2019.

[87] Y. Dong, Z. Hu, K. Wang, Y. Sun, and J. Tang, "Heterogeneous network representation learning," in *IJCAI*, 2020.

[88] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *arXiv preprint arXiv:1901.00596*, 2019.

[89] V. P. Dwivedi, C. K. Joshi, T. Laurent, Y. Bengio, and X. Bresson, "Benchmarking graph neural networks," *arXiv preprint arXiv:2003.00982*, 2020.

[90] H. Jiang, Y. Song, C. Wang, M. Zhang, and Y. Sun, "Semi-supervised learning over heterogeneous information networks by ensemble of meta-graph guided random walks." in *IJCAI*, 2017.

[91] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: Homophily in social networks," *Annual review of sociology*, vol. 27, no. 1, pp. 415–444, 2001.

[92] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[93] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[94] M. Belkin and P. Niyogi, “Laplacian eigenmaps and spectral techniques for embedding and clustering,” in *NIPS*, 2002.

[95] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, “Learning with local and global consistency,” in *NIPS*, 2004.

[96] X. Zhu, Z. Ghahramani, J. Lafferty *et al.*, “Semi-supervised learning using gaussian fields and harmonic functions,” in *ICML*, 2003.

[97] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in *NIPS*, 2013.

[98] H. Zhang, L. Qiu, L. Yi, and Y. Song, “Scalable multiplex network embedding.” in *IJCAI*, 2018.

[99] Y. He, Y. Song, J. Li, C. Ji, J. Peng, and H. Peng, “Hetespaceywalk: a heterogeneous spacey random walk for heterogeneous information network embedding,” in *CIKM*, 2019, pp. 639–648.

[100] C. Park, D. Kim, Q. Zhu, J. Han, and H. Yu, “Task-guided pair embedding in heterogeneous network,” in *CIKM*, 2019, pp. 489–498.

[101] J. Tang, M. Qu, and Q. Mei, “Pte: Predictive text embedding through large-scale heterogeneous text networks,” in *KDD*, 2015.

[102] M. Qu, J. Tang, J. Shang, X. Ren, M. Zhang, and J. Han, “An attention-based collaboration framework for multi-view network representation learning,” in *CIKM*, 2017.

[103] W. Zhang, Y. Fang, Z. Liu, M. Wu, and X. Zhang, “mg2vec: Learning relationship-preserving heterogeneous graph representations via metagraph embedding,” *IEEE TKDE*, vol. 14, no. 8, p. 1, 2020.

[104] J. Qiu, Y. Dong, H. Ma, J. Li, K. Wang, and J. Tang, “Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec,” in *WSDM*, 2018, pp. 459–467.

[105] H. Wang, F. Zhang, M. Hou, X. Xie, M. Guo, and Q. Liu, “Shine: Signed heterogeneous information network embedding for sentiment link prediction,” in *WSDM*, 2018.

[106] A. M. Fard, E. Bagheri, and K. Wang, “Relationship prediction in dynamic heterogeneous information networks,” in *ECIR*, 2019, pp. 19–34.

[107] V. W. Zheng, M. Sha, Y. Li, H. Yang, Y. Fang, Z. Zhang, K.-L. Tan, and K. C.-C. Chang, “Heterogeneous embedding propagation for large-scale e-commerce user alignment,” in *ICDM*, 2018.

[108] X. Fu, J. Zhang, Z. Meng, and I. King, “Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding,” in *WWW*, 2020, pp. 2331–2341.

[109] H. Tong, C. Faloutsos, and J.-Y. Pan, “Fast random walk with restart and its applications,” in *ICDM*, 2006.

[110] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *NIPS*, 2017, pp. 5998–6008.

[111] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” in *NAACL*, 2019, pp. 4171–4186.

[112] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[113] Z. Wang, J. Zhang, J. Feng, and Z. Chen, “Knowledge graph embedding by translating on hyperplanes,” in *AAAI*, 2014.

[114] Z. Sun, Z. Deng, J. Nie, and J. Tang, “Rotate: Knowledge graph embedding by relational rotation in complex space,” in *ICLR*, 2019.

[115] M. Nickel, V. Tresp, and H.-P. Kriegel, “A three-way model for collective learning on multi-relational data.” in *ICML*, 2011.

[116] J. Qiu, H. Ma, Y. Dong, K. Wang, and J. Tang, “Revisiting knowledge base embedding as tensor decomposition,” 2018. [Online]. Available: https://openreview.net/pdf?id=S1sRrN-CW

[117] J. Shang, J. Liu, M. Jiang, X. Ren, C. R. Voss, and J. Han, “Automated phrase mining from massive text corpora,” *TKDE*, vol. 30, no. 10, pp. 1825–1837, 2018.

[118] X. Wang, Y. Zhang, Q. Li, X. Ren, J. Shang, and J. Han, “Distantly supervised biomedical named entity recognition with dictionary expansion,” in *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, 2019, pp. 496–503.

[119] Q. Li, X. Wang, Y. Zhang, F. Ling, C. H. Wu, and J. Han, “Pattern discovery for wide-window open information extraction in biomedical literature,” in *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, 2018, pp. 420–427.

[120] F. M. Suchanek, G. Kasneci, and G. Weikum, “Yago: a core of semantic knowledge,” in *Proceedings of the 16th international conference on World Wide Web*, 2007, pp. 697–706.

[121] D. Vrandečić and M. Krötzsch, “Wikidata: a free collaborative knowledgebase,” *Communications of the ACM*, vol. 57, no. 10, pp. 78–85, 2014.

[122] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, “Liblinear: A library for large linear classification,” in *ICML*, 2008.

[123] M. Wang, L. Yu, D. Zheng, Q. Gan, Y. Gai, Z. Ye, M. Li, J. Zhou, Q. Huang, C. Ma, Z. Huang, Q. Guo, H. Zhang, H. Lin, J. Zhao, J. Li, A. J. Smola, and Z. Zhang, “Deep graph library: Towards efficient and scalable deep learning on graphs,” *ICLRW*, 2019.

[124] M. Fey and J. E. Lenssen, “Fast graph representation learning with PyTorch Geometric,” in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[125] F. Zhang, X. Liu, J. Tang, Y. Dong, P. Yao, J. Zhang, X. Gu, Y. Wang, B. Shao, R. Li *et al.*, “Oag: Toward linking large-scale heterogeneous entity graphs,” in *KDD*, 2019, pp. 2585–2595.

[126] X. Han, S. Cao, L. Xin, Y. Lin, Z. Liu, M. Sun, and J. Li, “Openke: An open toolkit for knowledge embedding,” in *Proceedings of EMNLP*, 2018.