A Survey on Heterogeneous Graph Embedding: Methods, Techniques, Applications and Sources

Xiao Wang, Deyu Bo, Chuan Shi†, *Member, IEEE,* Shaohua Fan, Yanfang Ye, *Member, IEEE,* and Philip S. Yu, *Fellow, IEEE*

Abstract—Heterogeneous graphs (HGs) also known as heterogeneous information networks have become ubiquitous in real-world scenarios; therefore, HG embedding, which aims to learn representations in a lower-dimension space while preserving the heterogeneous structures and semantics for downstream tasks (e.g., node/graph classification, node clustering, link prediction), has drawn considerable attentions in recent years. In this survey, we perform a comprehensive review of the recent development on HG embedding methods and techniques. We first introduce the basic concepts of HG and discuss the unique challenges brought by the heterogeneity for HG embedding in comparison with homogeneous graph representation learning; and then we systemically survey and categorize the state-of-the-art HG embedding methods based on the information they used in the learning process to address the challenges posed by the HG heterogeneity. In particular, for each representative HG embedding method, we provide detailed introduction and further analyze its pros and cons; meanwhile, we also explore the transformativeness and applicability of different types of HG embedding methods in the real-world industrial environments for the first time. In addition, we further present several widely deployed systems that have demonstrated the success of HG embedding techniques in resolving real-world application problems with broader impacts. To facilitate future research and applications in this area, we also summarize the open-source code, existing graph learning platforms and benchmark datasets. Finally, we explore the additional issues and challenges of HG embedding and forecast the future research directions in this field.

Index Terms—heterogeneous graph, graph embedding, machine learning, deep learning.

1 Introduction

Heterogeneous graphs (HGs) [1], which are capable of composing different types of entities (i.e., nodes) and relations, also known as heterogeneous information network, have become ubiquitous in real world scenarios, ranging from bibliographic networks, social networks to recommendation systems. For example, as shown in Fig. 1(a), a bibliographic network (i.e., academic network) can be represented by a HG, which consists of four types of nodes (author, paper, venue, and term) and three types of edges (author-write-paper, paper-contain-term and conference-publish-paper); and these basic relations can be further derived for more complex semantics over the HG (e.g., author-write-paper-contain-item). It has been well recognized that HG is a powerful model that is able to embrace rich semantics and structural information in real world data. Therefore, researches on HG data have been experiencing tremendous growth in data mining and machine learning, many of which have been successfully applied in real world systems such as recommendation [2], [3], text analysis [4], [5], and cybersecurity [6], [7].

Due to the ubiquity of HG data, how to learn embeddings of HG is a key research problem in various graph analysis applications, e.g., node/graph classification [8], [9], and node clustering [10]. Traditionally, to learn HG embeddings, matrix (e.g., adjacency matrix) factorization methods [11], [12] have been proposed to generate latent-dimension features in a HG. However, the computational cost of decomposing a large-scale matrix is usually very expensive, and also suffers from its statistical performance drawback [13], [14]. To address this challenge, heterogeneous graph embedding (i.e., heterogeneous graph representation learning), aiming to learn a function that maps input space into a lower-dimension space while preserving the heterogeneous structure and semantics, has drawn considerable attentions in recent years. Although there have been ample studies of embedding technology on homogeneous graphs [14] which consist of only one type of nodes and edges, these techniques cannot be directly applicable to HGs due to the heterogeneity of HG data. More specifically, i) the structure in HG is usually semantic dependent, e.g., meta-path structure [8], implying that the local structure of one node in HG can be very different when considering different types of relations; ii) different types of nodes and edges have different attributes, which are usually located in different feature spaces, and thus when designing heterogeneous graph embedding methods, especially heterogeneous graph neural networks (HGNNs), we need to overcome the heterogeneity of attributes to fuse information [15], [16]; iii) another one is that HG is usually application dependent: for example, the basic structure of HG usually can be captured by meta-path, however meta-path selection is still challenging in reality, which may need sufficient domain knowledge. To tackle the above issues, various heteroge-

† *Corresponding author*

- *X. Wang, D. Bo, C. Shi and S. Fan are with the Beijing Key Lab of Intelligent Telecommunications Software and Multimedia, Beijing University of Posts and Telecommunications, Beijing 100876, China. E-mail: {xiaowang, bodeyu, shichuan, fanshaohua}@bupt.edu.cn.*
- *Y. Ye is with the Department of Computer and Data Sciences, Case Western Reserve University, Cleveland, OH, 44106. E-mail: yanfang.ye@case.edu.*
- *P. S. Yu is with Computer Science Department, University of Illinois at Chicago, Chicago, IL 60607, and also with the Institute for Data Science, Tsinghua University, Beijing 100084, China. E-mail: psyu@uic.edu.*

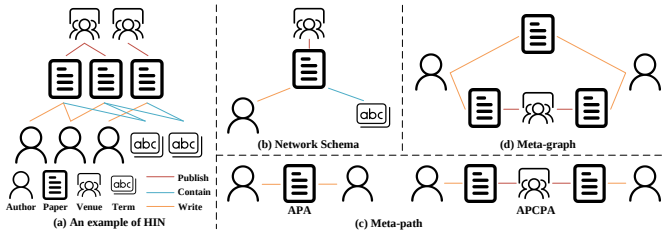


Fig. 1: An illustrative example of a heterogeneous graph. (a) An academic network including four types of node (i.e., Author, Paper, Venue, Term) and three types of link (i.e., Publish, Contain, Write). (b) Network schema of the academic network. (c) Two meta-paths used in the academic network (i.e., Author-Paper-Author and Paper-Term-Paper). (d) A meta-graph used in the academic network.

neous graph embedding methods have been proposed [17], [18], [8], [9], [15], [2], many of which [19], [20], [6], [21], [22], [23] have demonstrated the success of heterogeneous graph embedding techniques deployed in real world applications including recommendation systems [2], [3], malware detection systems [7], [22], [23], [24], and healthcare systems [25], [26].

Although ample studies of heterogeneous graph embedding have been conducted with various applications in different fields, there have not been systematic and comprehensive survey on heterogeneous graph embedding methods with in-depth analysis of their pros and cons and detailed discussion of their transformativeness and applicability. To bridge this gap, in this paper, we will thoroughly survey the existing works on heterogeneous graph embedding, including representative methods and techniques, deployed systems in real world applications, and publicly available benchmark datasets as well as open-source code/tools. In particular, (1) we will explore recent progress of heterogeneous graph embedding, by introducing its representative methods and techniques with analysis of their pros and cons; then (2) we will introduce and discuss the transformativeness of existing heterogeneous graph embedding methods that have been successfully deployed in real-world applications; afterwards (3) we will summarize publicly available benchmark datasets and open-source code/tools to facilitate researchers and practitioners for future heterogeneous graph embedding works; and finally (4) we will discuss the additional issues and challenges of heterogeneous graph embedding technique and forecast the future research directions in this area. Note that different from the existing surveys which mainly focus on homogeneous graph embedding [14], [27], [28], [29], [30], [31], we aim at exploring the works on heterogeneous graph embedding. Although there have been few survey works on heterogeneous graph embedding [32], [33], we make our unique contributions in this work as summarized below.

- We first discuss the unique challenges brought by the heterogeneity of HG compared with homogeneous graphs; and then we provide a comprehensive survey of existing heterogeneous graph embedding methods, which are categorized based on the information they used in the learning process to address particular type of challenges posed by the HG heterogeneity.
- For each representative heterogeneous graph embedding method and technique, we provide detailed introduction and further analyze its pros and cons. In addition, we explore the transformativeness and applicability of different types of HG embedding methods in the real-world industrial environments for the first time.
- We summarize the open-source code and benchmark datasets, and give a detailed description to the existing graph learning platforms, to facilitate future research and applications in this area.
- We also explore the additional issues and challenges of heterogeneous graph embedding and forecast the future research directions in this field.

The remainder of this survey paper is organized as follows. In Section 2, we first introduce the HG concepts and discuss the unique challenges of heterogeneous graph embedding due to the heterogeneity. In Section 3, we categorize and introduce heterogeneous graph embedding methods in details according to the information (e.g., structures, attributes, and application dependent domain knowledge) used in the learning process, based on which we analyze their pros and cons and then discuss their applicability. In Section 4, we further summarize the commonly used techniques in the state-of-the-art heterogeneous graph embedding methods. In Section 5, we further explore the transformativeness of existing heterogeneous graph embedding methods that have been successfully deployed in real-world application systems. Section 5 summarizes the benchmark datasets and open-source code/tools for heterogeneous graph embedding. Section 7 discusses additional issues/challenges of heterogeneous graph embedding and forecasts the future research directions in this field.

2 Preliminary

2.1 Basic Concepts

HG is a graph consisting of different types of entities (i.e., nodes) and/or different types of relations (i.e., edges), which can be defined as follows.

Definition 1. **Heterogeneous graph (or heterogeneous information network)** [1]. A HG is defined as a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, in which $\mathcal{V}$ and $\mathcal{E}$ represent the node set and the link set, respectively. Each node $v \in \mathcal{V}$ and each link $e \in \mathcal{E}$ are associated with their mapping function $\phi(v) : \mathcal{V} \rightarrow \mathcal{A}$ and $\varphi(e) : \mathcal{E} \rightarrow \mathcal{R}$. $\mathcal{A}$ and $\mathcal{R}$ denote the node types and link types, respectively, where $\mathcal{A} + \mathcal{R} > 2$. The **network schema** for $\mathcal{G}$ is defined as $\mathcal{S} = (\mathcal{A}, \mathcal{R})$, which can be seen as a meta template of a heterogeneous graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with the node type mapping function $\phi(v) : \mathcal{V} \rightarrow \mathcal{A}$ and the link type mapping function $\varphi(e) : \mathcal{E} \rightarrow \mathcal{R}$. The network schema is a graph defined over node types $\mathcal{A}$, with links as relations from $\mathcal{R}$.

HG not only provides the graph structure of the data associations, but also provides a higher-level semantics of the data. An example of HG is illustrated in Fig. 1(a), which consists of four node types (author, paper, venue, and term) and three link types (author-write-paper, paper-contain-term, and conference-publish-paper); while Fig. 1(b)

TABLE 1: Notations and Explanations

Notations	Explanations
d	dimension of node embeddings
N	Number of nodes
m	Meta-path
$\mathbf{h}_i$	Attributes or embeddings of node i
$\mathbf{M}_r$	Relation-specific matrix of relation r
w_{ij}	Weight of link between node i and node j
S_r	Heterogeneous similarity function with relation r
$C_t(i)$	Context nodes of node i with type t
$\mathcal{N}_i$	Neighbors of node i
σ	Sigmoid function
$\odot$	Hadamard product
$\oplus$	Concatenation operator

illustrates the network schema. Based on a constructed HG, to formulate the semantics of higher-order relationships among entities, meta-path [34] is further proposed whose definition is given below.

Definition 2. **Meta-path** [34]. A meta-path m is based on a network schema $\mathcal{S}$, which is denoted as $m = A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \cdots \xrightarrow{R_l} A_{l+1}$ (simplified to $A_1A_2\cdots A_{l+1}$) with node types $A_1, A_2, \cdots, A_{l+1} \in \mathcal{A}$ and link types $R_1, R_2, \cdots R_l \in \mathcal{R}$.

Note that different meta-paths describe semantic relationships in different views. For example, the meta-path of "APA" indicates the co-author relationship and "APCPA" represents the co-conference relation. Both of them can be used to formulate the relatedness over authors. Although meta-path can be used to depict the relatedness over entities, it fails to capture a more complex relationship, such as motifs [35]. To address this challenge, meta-graph [36] is proposed to use a directed acyclic graph of entity and relation types to capture more complex relationship between two HG entities, defined as follows.

Definition 3. **Meta-graph** [36]. A meta-graph $\mathcal{T}$ can be seen as a directed acyclic graph (DAG) composed of multiple meta-paths with common nodes. Formally, meta-graph is defined as $\mathcal{T} = (V_\mathcal{T}, E_\mathcal{T})$, where $V_\mathcal{T}$ is a set of nodes and $E_\mathcal{T}$ is a set of links. For any node $v \in V_\mathcal{T}, \phi(v) \in \mathcal{A}$; for any link $e \in E_\mathcal{T}, \varphi(e) \in \mathcal{R}$.

An example meta-graph is shown in Fig. 1(d), which can be regarded as the combination of meta-path "APA" and "APCPA", reflecting a high-order similarity of two nodes. Note that a meta-graph can be symmetric or asymmetric [37]. To learn embeddings of HG data, we formalize the problem of heterogeneous graph embedding as follow.

Definition 4. **Heterogeneous graph embedding** [13]. Heterogeneous graph embedding aims to learn a function $\Phi : \mathcal{V} \to \mathbb{R}^d$ that embeds the nodes $v \in \mathcal{V}$ in HG into a low-dimensional Euclidean space with $d \ll |\mathcal{V}|$.

Table 1 summarizes symbols used through this paper.

2.2 Challenges of HG Embedding due to Heterogeneity

Different from homogeneous graph embedding [14], where the basic problem is preserving structure and property in node embedding [14]. Due to the heterogeneity, heterogeneous graph embedding imposes more challenges, which are illustrated below.

- **Complex structure** (the complex HG structure caused by multiple types of nodes and edges). In a homogeneous graph, the fundamental structure can be considered as the so-called first-order, second-order, and even higher-order structure [38], [39], [40]. All these structures are well defined and have good intuition. However, the structure in HG will dramatically change depending on the selected relations. Let's still take the academic network in Fig. 1(a) as an example, the neighbors of one paper will be authors with the "write" relation, while with "contain" relation, the neighbors become terms. Complicating things further, the combination of these relations, which can be considered as a higher-order structure in HG, will result in different and more complicated structures. Therefore, how to efficiently and effectively preserve these complex structures is of great challenge in heterogeneous graph embedding, while current efforts have been made towards the meta-path structure [8] and meta-graph structure [41], etc.
- **Heterogeneous attributes** (the fusion problem caused by the heterogeneity of attributes). Since the nodes and edges in a homogeneous graph have the same type, each dimension of the node or edge attributes has the same meaning. In this situation, node can directly fuse the attributes of its neighbors. However, in heterogeneous graph, the attributes of different types of nodes and edges may have different meanings [16], [15]. For example, the attributes of author can be the research fields, while paper may use keywords as attributes. Therefore, how to overcome the heterogeneity of attributes and effectively fuse the attributes of neighbors poses another challenge in heterogeneous graph embedding.
- **Application dependent**. HG is closely related to the real world applications, while many practical problems remain unsolved. For example, constructing an appropriate HG may require sufficient domain knowledge in a real world application. Also, meta-path and/or meta-graph are widely used to capture the structure of HG. However, unlike homogeneous graph, where the structure (e.g., the first-order and second-order structure) is well defined, meta-path selection may also need prior knowledge. Furthermore, to better facilitate the real world applications, we usually need to elaborately encode the side information (e.g., node attributes) [15], [16], [42], [43] or more advanced domain knowledge [2], [44], [45] to the heterogeneous graph embedding process.

3 Method Taxonomy

Various types of nodes and links in HG bring different graph structures and rich attributes (i.e., heterogeneity). As discussed in Section 2.2, in order to make the node embeddings capture the heterogeneous structures and rich attributes, we need to consider the information of different aspects in the embedding, including graph structures, attributes and specific application labels, etc. Based on the aforementioned challenges, in this section, we categorize the existing methods into four categories based on the information they used in heterogeneous graph embedding: (1) *Structure-preserved heterogeneous graph embedding*. The methods belonging to this category primarily focus on

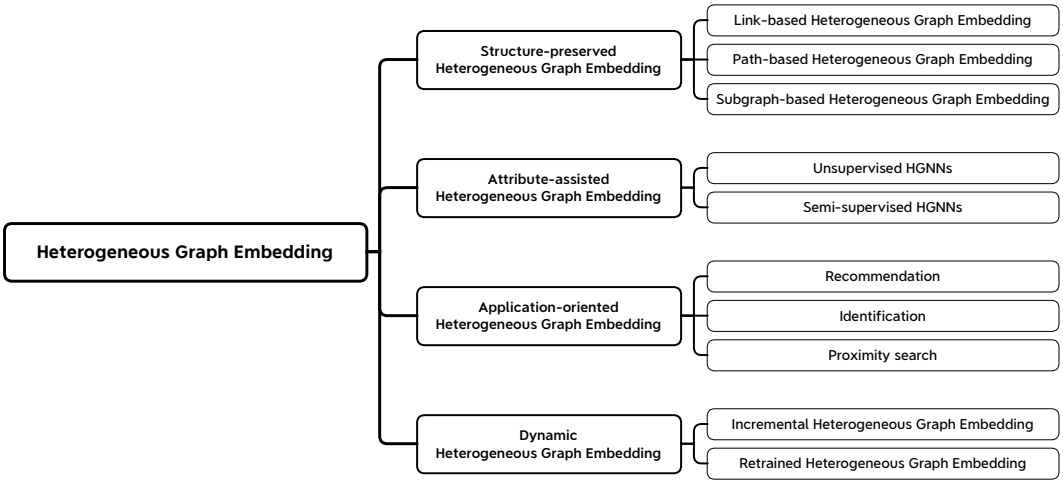


Fig. 2: An overview of heterogeneous graph embedding from the perspective of used information.

capturing and preserving the heterogeneous structures and semantics, e.g., the meta-path and meta-graph. (2) *Attribute-assisted heterogeneous graph embedding*. The methods incorporate more information beyond structure, e.g., node and edge attributes, into embedding technology, so as to utilize the neighborhood information more effectively. (3) *Application-oriented heterogeneous graph embedding*. We further explore the applicability of the heterogeneous graph embedding methods (i.e., the ones aim to learn application-oriented node embeddings over HG). (4) *Dynamic heterogeneous graph embedding*. Different from existing survey works that mainly focus on the embedding methods for static heterogeneous graphs. In this work, we further explore and summarize dynamic heterogeneous graph embedding methods, which aim to capture the evolution of heterogeneous graphs and preserve the temporal information in the node embeddings. An overview of different types of HG embedding methods explored in this survey paper is shown in Fig. 2.

3.1 Structure-preserved HG Embedding

One basic requirement of graph embedding is to preserve the graph structures properly [14]. Accordingly, the homogeneous graph embedding pays more attention on higher-order graph structures, for example, second-order structures [39], [46], high-order structures [47], [48] and community structures [40]. However, one typical characteristic of HG is that it contains multiple relations among nodes, which inevitably needs to consider the heterogeneity of graph. Therefore, an important direction of heterogeneous graph embedding is to learn semantic information from the graph structures. In this section, we review the typical heterogeneous graph embedding methods based on the basic HG structures, including link (i.e., edge), meta-path, and subgraph. Link is the observed relation between two nodes, meta-path is composed of different types of links and subgraph represents the tiny sub-structure of graph. The three structures are the most fundamental ingredients of HG, which are able to capture the semantic information from different perspectives. In the followings, we will review the typical structure-preserved heterogeneous graph embedding methods based on these three types of structures and then discuss their pros and cons.

3.1.1 Link-based HG Embedding

One of the most basic information that heterogeneous graph embedding needs to preserve is link. Different from homogeneous graph, link in HG has different types and contains different semantics. To distinguish various types of links, one classical idea is to deal with them in different metric spaces, rather than a unified metric space. A representative work is PME [17], which treats each link type as a relation and uses a relation-specific matrix to transform the nodes into different metric spaces. In this way, nodes connected by different types of links can be close to each other in different metric spaces, thus capturing the heterogeneity of the graph. The distance function is defined as follows:

$$S_r(v_i, v_j) = w_{ij} \left\| \mathbf{M}_r \mathbf{h}_i - \mathbf{M}_r \mathbf{h}_j \right\|_2, \tag{1}$$

where $\mathbf{h}_i$ and $\mathbf{h}_j \in \mathbb{R}^{d\times 1}$ denote the node embeddings of node i and node j, respectively; $\mathbf{M}_r \in \mathbb{R}^{d\times d}$ is the projection matrix of relation r; and w_{ij} represents the weight of link between node i and node j. Note that Eq. 1 can be seen as a metric learning function:

$$\left\| \mathbf{M}_r(\mathbf{h}_i - \mathbf{h}_j) \right\|_2 = \sqrt{(\mathbf{h}_i - \mathbf{h}_j)^\top \mathbf{M}_r^\top \mathbf{M}_r (\mathbf{h}_i - \mathbf{h}_j)}, \tag{2}$$

where $\mathbf{M}_r^\top \mathbf{M}_r \in \mathbb{R}^{d\times d}$ is the metric matrix of Mahalanobis distance [49]. PME considers the relations between nodes when minimizing the distance of them, thus capturing the heterogeneity of graph. The loss function is the margin-based triple loss function, which requires a distance between the positive and negative samples:

$$L = \sum_{r\in\mathcal{R}} \sum_{(v_i,v_j)\in E_r} \sum_{(v_i,v_k)\notin E_r} [\xi + S_r(v_i, v_j)^2 - S_r(v_i, v_k)^2]_+ \tag{3}$$

where ξ denotes the margin, E_r represents the positive links of relation r, and $[z]_+ = \max(z, 0)$. Through Eq. 3, PME makes the node pairs connected by relation r closer to each other than the node pairs without relation r.

By exploiting the relation-specific matrix to capture the link heterogeneity, different from PME, other methods have

been proposed aiming to maximize the similarity of two nodes connected by specific relations. For example, EOE [50] and HeGAN [18] use the relation-specific matrix $\mathbf{M}_r$ to calculate the similarity between two nodes, which can be formulated as:

$$S_r(v_i, v_j) = \frac{1}{1 + \exp\left\{-\mathbf{h}_i^\top \mathbf{M}_r \mathbf{h}_j\right\}}. \quad (4)$$

More specifically, EOE is proposed to learn embeddings for coupled heterogeneous graphs, which consist of two different but related sub-graphs. It divides the links in HG into intra-graph links and inter-graph links. Intra-graph link connects two nodes with the same type, and inter-graph link connects two nodes with different types. To capture the heterogeneity in inter-graph link, EOE utilizes Eq. 4 as the similarity function of two nodes. Different from EOE, HeGAN uses generative adversarial networks (GAN) [51] to learn node embeddings for heterogeneous graph. It uses Eq. 4 as a discriminator to determine whether the node embeddings are produced by the generator. Through the game between discriminator and generator, HeGAN can learn more robust node embeddings.

The previously discussed methods mainly preserve the link structure based on either the distance or similarity function on node embeddings, while AspEM [52] and HEER [53] aim to maximize the probability of existing links. The heterogeneous similarity function is defined as:

$$S_r = \frac{\exp(\boldsymbol{\mu}_r^\top \mathbf{g}_{ij})}{\sum_{\tilde{i} \in E_{\tilde{i}j}^r} \exp(\boldsymbol{\mu}_r^\top \mathbf{g}_{\tilde{i}j}) + \sum_{\tilde{j} \in E_{i\tilde{j}}^r} \exp(\boldsymbol{\mu}_r^\top \mathbf{g}_{i\tilde{j}})}, \quad (5)$$

where $\boldsymbol{\mu}_r \in \mathbb{R}^{d \times 1}$ is the embedding of relation r; $\mathbf{g}_{ij} \in \mathbb{R}^{d \times 1}$ is the embedding of link between node i and node j; $\mathbf{g}_{ij} = \mathbf{h}_i \odot \mathbf{h}_j$ and $\odot$ denotes the Hadamard product; and $E_{\tilde{i}j}^r$ is the set of negative links, which indicates that there is no link between node $\tilde{i}$ and node j. It can be seen that $\boldsymbol{\mu}_r^\top \mathbf{g}_{ij}$ measures the closeness between link and its corresponding type. Maximizing S_r enlarges the closeness between the existing links and their corresponding types, thus capturing the heterogeneity of the graph.

In addition to the above methods, there are some methods that draw on techniques from other fields. Similar to the idea of TransE [54], MELL [55] uses the equation 'head + relation = tail' to learn the node embeddings for heterogeneous graph. PTE [56] decomposes the heterogeneous graph into multiple bipartite graphs and employs LINE [39], which preserves the first- and second-order structures of graph, to learn node embeddings for each bipartite graph. MNE [57] assigns multiple embeddings for each node and uses a skip-gram technique [58] to represent information of multi-type relations into a unified space.

In summary, we can roughly divide the link-based heterogeneous graph embedding methods into two categories: one is to explicitly preserve the proximity of links [52], [53]; the other is to preserve the proximity of nodes, which utilizes the information of links implicitly [17], [50], [18]. These two types of methods both make use of the first-order information of HG.

3.1.2 Path-based HG Embedding

Link-based methods can only capture the local structures of HG, i.e., the first-order relation. In fact, the higher-order relation, describing more complex semantic information, is also critical for heterogeneous graph embedding. For example, in Fig. 1(a), the first-order relation can only reflect the similarity of author-paper, paper-term and paper-venue. While the similarity of author-author, paper-paper and author-conference cannot be well captured. Therefore, the high-order relation is introduced to measure more complex similarity. Because the number of high-order relations is very large, in order to reduce complexity, we usually choose the higher-order relations with rich semantics, called meta-path. In this section, we will introduce some representative meta-path-based heterogeneous graph embedding methods, which can be divided into two categories: random walk-based methods [8], [59], [60], [61], [62] and hybrid relation-based methods [9], [63].

Random walk-based methods usually use meta-path to guide random walk on a HG, so that the generated node sequence contains rich semantic information. Through preserving the node sequence structure, node embedding can preserve both first-order and high-order proximity [8]. A representative work is metapath2vec [8] (shown in Fig. 3).

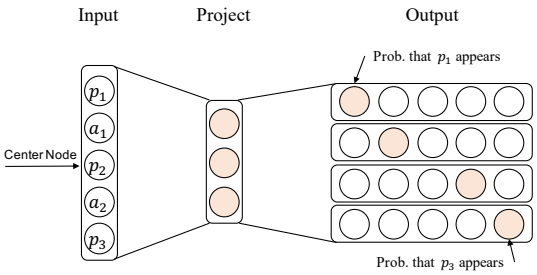


Fig. 3: The architecture of metapath2vec. Node sequence is generated under the meta-path PAP. It projects the embedding of the center node, e.g., p_2 into latent space and maximizes the probability of its meta-path-based context nodes, e.g., p_1, p_3, a_1 and a_2, appearing.

Metapath2vec [8] mainly uses meta-path guided random walk to generate heterogeneous node sequences with rich semantics; and then it designs a heterogeneous skip-gram technique to preserve the proximity between node v and its context nodes, i.e., neighbors in the random walk sequences:

$$\arg\max_\theta \sum_{v \in \mathcal{V}} \sum_{t \in \mathcal{A}} \sum_{c_t \in C_t(v)} \log p(c_t | v; \theta), \quad (6)$$

where $C_t(v)$ represents the context nodes of node v with type t. $p(c_t|v;\theta)$ denotes the heterogeneous similarity function on node v and its context neighbors c_t:

$$p(c_t | v; \theta) = \frac{e^{\mathbf{h}_{c_t} \cdot \mathbf{h}_v}}{\sum_{\tilde{v} \in \mathcal{V}} e^{\mathbf{h}_{\tilde{v}} \cdot \mathbf{h}_v}}, \quad (7)$$

From the diagram shown in Fig. 3, Eq. 7 needs to calculate the similarity between center node and its neighbors. Then [58] introduces a negative sampling strategy to reduce the computation. Hence, Eq. 7 can be approximated as:

$$\log \sigma(\mathbf{h}_{c_t} \cdot \mathbf{h}_v) + \sum_{q=1}^{Q} \mathbb{E}_{\tilde{v}^q \sim P(\tilde{v})} \left[\log \sigma\left(-\mathbf{h}_{\tilde{v}^q} \cdot \mathbf{h}_v\right)\right], \quad (8)$$

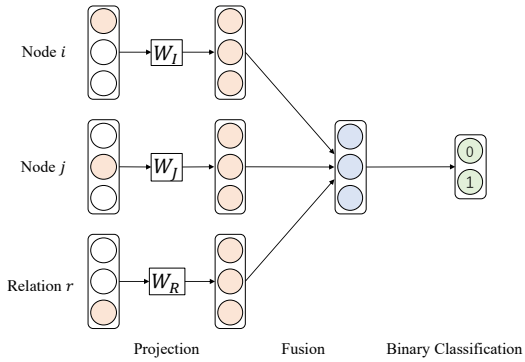


Fig. 4: The architecture of HIN2vec. Through the parameter matrices $\mathbf{W}_I, \mathbf{W}_J$ and $\mathbf{W}_R$, the one-hot vectors of node i, node j and relation r are projected into dense vectors. And these three vectors are fused by a neural network to predict whether node i and j are connected by relation r.

where $\sigma(\cdot)$ is the sigmoid function, and $P(\tilde{v})$ is the distribution in which the negative node $\tilde{v}^q$ is sampled for Q times. However, when choosing the negative samples, metapath2vec does not consider the types of nodes, i.e., different types of nodes are from the same distribution $P(\tilde{v})$. It further designs metapath2vec++, which samples the negative nodes of the same type as the central node. After minimizing the objective function, metapath2vec and metapath2vec++ can capture both structural information and semantic information effectively and efficiently.

Based on metapath2vec, a series of variants have been proposed. Spacey [59] designs a heterogeneous spacey random walk to unify different meta-paths with a second-order hyper-matrix to control the transition probability among different node types. JUST [60] proposes a random walk method with Jump and Stay strategies, which can flexibly choose to change or maintain the type of the next node in the random walk without meta-path. BHIN2vec [61] proposes an extended skip-gram technique to balance the various types of relations. It treats heterogeneous graph embedding as a multiple relation-based tasks, and balances the influence of different relations on node embeddings by adjusting the training ratio of different tasks. HHNE [62] conducts the meta-path guided random walk in hyperbolic spaces [64], where the similarity between nodes can be measured using hyperbolic distance. In this way, some properties of HG, e.g., hierarchical and power-law structure, can be naturally reflected in the learned node embeddings.

Different from random walk-based methods that learn structural and semantic information from generated node sequences, some methods use the combination of first-order relation and high-order relation (i.e., meta-path) to capture the heterogeneity of HG. We call these work as hybrid relation-based methods. A typical work is HIN2vec [9] (shown in Fig. 4), which carries out multiple relation prediction tasks jointly to learn the embeddings of nodes and meta-paths.

The purpose of HIN2vec is to predict whether two nodes are connected by a meta-path, which can be seen as a multi-label classification task. As illustrated in Fig. 4, given two nodes i and j, HIN2vec uses the following function to compute their similarity under the hybrid relation r:

$$S_r(v_i, v_j) = \sigma(\sum \mathbf{W}_I \vec{i} \odot \mathbf{W}_J \vec{j} \odot f_{01}(\mathbf{W}_R \vec{r})), \quad (9)$$

where $\vec{i}, \vec{j}$ and $\vec{r} \in \mathbb{R}^{N\times 1}$ denote the one-hot vectors of nodes and relation, respectively; $\mathbf{W}_I, \mathbf{W}_J$ and $\mathbf{W}_R \in \mathbb{R}^{d\times N}$ are the mapping matrices; and $f_{01}(\cdot)$ is a regularization function, which limits the embedding values between 0 and 1. The loss function is a binary cross-entropy loss:

$$E^r_{ij} \log S_r(v_i, v_j) + [1 - E^r_{ij}] \log[1 - S_r(v_i, v_j)], \quad (10)$$

where E^r_{ij} denotes the set of positive links. After minimizing the loss function, HIN2vec can learn the embedding of nodes and relations (meta-paths). Besides, in the relation set R, it contains not only the first-order structures (e.g., A-P relation) but also the high-order structures (e.g., A-P-A relation). Therefore, the node embeddings can capture different semantics.

RHINE [63] is another hybrid relation-based method, which designs different distance functions for different relations, thus enhancing the expressive power of node embeddings. It divides the relations into two categories: Affiliation Relations (ARs) and Interaction Relations (IRs). For ARs, a Euclidean distance function is introduced; while for IRs, RHINE proposes a translation-based distance function. Through the combination of these two distance functions, RHINE can learn relation structure-aware heterogeneous node embeddings.

In sum, we can find that random walk-based methods mainly exploit meta-path guided strategy for heterogeneous graph embedding; while hybrid relation-based methods regard a meta-path as high-order relation and learn meta-path based embeddings simultaneously. Compared with random walk-based methods, hybrid relation-based methods can simultaneously integrate multiple meta-paths into heterogeneous graph embedding flexibly.

3.1.3 Subgraph-based HG Embedding

Subgraph represents a more complex structure in the graph. Incorporating subgraphs into graph embedding can significantly improve the ability of capturing complex structural relationships. In this section, we introduce two widely used subgraphs in HG: one is metagraph, which reflects the high-order similarity between nodes [41], [37]; the other is the hyperedge[1], which connects a series of closely related nodes and preserves the indecomposablity among them [65].

Zhang *et al.* propose metagraph2vec [41], which uses a metagraph-guided random walk to generate heterogeneous node sequence. Then the heterogeneous skip-gram technique [8] is employed to learn the node embeddings. Based on this strategy, metagraph2vec can capture the rich structural information and high-order similarity among nodes. Different from metagraph2vec that only uses metagraphs in the pre-processing step (i.e., metagraph-guided random walk), mg2vec [37] aims to learn the embeddings for metagraphs and nodes jointly, so that the metagraphs can join the learning process. It first enumerates metagraphs and then preserves the proximity between nodes and metagraphs:

$$P(\mathbf{M}_i|v) = \frac{\exp(\mathbf{M}_i \cdot \mathbf{h}_v)}{\sum_{M_j \in \mathcal{M}} \exp(\mathbf{M}_j \cdot \mathbf{h}_v)}, \quad (11)$$

1. In this paper, we treat the hyperedge as a special kind of subgraph.

where $\mathbf{M}_i$ is the embedding of metagraph i and $\mathcal{M}$ denotes the set of metagraphs. Clearly, $P(M_i|v)$ represents the first-order relationship between the nodes and its subgraphs. Further, mg2vec preserves the proximity between node pair and its subgraph to capture the second-order information:

$$P(\mathbf{M}_i|u,v) = \frac{\exp(\mathbf{M}_i \cdot f(\mathbf{h}_u, \mathbf{h}_v))}{\sum_{M_j \in \mathcal{M}} \exp(\mathbf{M}_j \cdot f(\mathbf{h}_u, \mathbf{h}_v))}, \tag{12}$$

where $f(\cdot)$ is a neural network to learn the embeddings of node pairs. Through preserving the first-order and second-order proximity between nodes and metagraphs, mg2vec can capture the structural information and the similarity between nodes and metagraphs.

DHNE [65] is a typical hyperedge-based graph embedding method. Specifically, it designs a novel deep model to produce a non-linear tuple-wise similarity function while capturing the local and global structures of a given HG. Taking a hyperedge with three nodes a, b and c as an example. The first layer of DHNE is an autoencoder, which is used to learn latent embeddings and preserve the second-order structures of graph [39]. The second layer is a fully connected layer with embedding concatenated:

$$\mathbf{L} = \sigma(\mathbf{W}_a\mathbf{h}_a \oplus \mathbf{W}_b\mathbf{h}_b \oplus \mathbf{W}_c\mathbf{h}_c), \tag{13}$$

where $\mathbf{L}$ denotes the embedding of the hyperedge; $\mathbf{h}_a, \mathbf{h}_b$ and $\mathbf{h}_c \in \mathbb{R}^{d\times 1}$ are the embeddings of node a, b and c learn by the autoencoder. $\mathbf{W}_a, \mathbf{W}_b$ and $\mathbf{W}_c \in \mathbb{R}^{d'\times d}$ are the transformation matrices for different node types. Finally, the third layer is used to calculate the indecomposability of the hyperedge:

$$\mathcal{P} = \sigma(\mathbf{W} * \mathbf{L} + \mathbf{b}), \tag{14}$$

where $\mathcal{P}$ denote the indecomposability of the hyperedge; $\mathbf{W} \in \mathbb{R}^{1\times 3d'}$ and $\mathbf{b} \in \mathbb{R}^{1\times 1}$ are the weight matrix and bias, respectively. A higher value of $\mathcal{P}$ means these nodes are from the existing hyperedges, otherwise it should be small. HEBE [66] is another hyperedge-based method, which aims to maximize the proximity between the node and the hyperedge it belongs to. After maximizing the proximity, HEBE can preserve the similarity of nodes within the same hyperedge, while reduce the similarity of nodes from different hyperedges. Besides, [67] proposes hyper-path-based random walk to preserve both the structural information and indecomposability of the hyper-graphs.

Compared with the structures of link and meta-path, subgraph (with two representative forms of meta-graph and hyperedge) usually contains much higher order structural and semantic information. However, one obstacle of subgraph-based heterogeneous graph embedding methods is the high complexity of subgraph. How to balance the effectiveness and efficiency is required for a practical subgraph-based heterogeneous graph embedding methods, which is worthy of further exploration.

3.1.4 Summary

Generally, structure-preserved heterogeneous graph embedding methods mainly use shallow models, i.e., models without non-linear activation and multiple transformation. A major advantage of this type of methods is that they have good parallelizability and can improve training speed through negative sampling [58]. However, as we can see, there has been increasingly advanced structural and semantic information from link to path to subgraph, which may improve the performance in nature, but it also requires more calculations. Besides, there are two serious problems: one is that the shallow models need to assign each node a low-dimensional embedding, which requires larger memory spaces to store the parameters. Another is that shallow models can only work on transductive setting, i.e., they cannot learn the embedding of new node. These two shortcomings limit the application of this kind of methods in large-scale industrial scenarios.

3.2 Attribute-assisted HG Embedding

In addition to the graph structures, another important component of heterogeneous graph embedding is the rich attributes. Attribute-assisted heterogeneous graph embedding methods aim to encode the complex structures and multiple attributes to learn node embeddings. Different from graph neural networks (GNNs) that can directly fuse the attributes of neighbors to update node embeddings, due to the different types of nodes and edges, HGNNs need to overcome the heterogeneity of attributes and design effective fusion methods to utilize the neighborhood information, thus bringing more challenges. In this section, we divide HGNNs into unsupervised and semi-supervised settings, then discuss their pros and cons.

3.2.1 Unsupervised HGNNs

Unsupervised HGNNs aim to learn node embeddings with good generalization. To this end, they always utilize the interactions among different types of attributes to capture the potential commonalities.

HetGNN [16] is the representative work of unsupervised HGNNs. It consists of three parts: content aggregation, neighbor aggregation and type aggregation. Content aggregation is designed to learn fused embeddings from different node contents, such as images, text or attributes:

$$f_1(v) = \frac{\sum_{i\in C_v}[\overrightarrow{LSTM}\{\mathcal{FC}(\mathbf{h}_i)\} \oplus \overleftarrow{LSTM}\{\mathcal{FC}(\mathbf{h}_i)\}]}{|C_v|}, \tag{15}$$

where C_v is the type of node v's attributes. $\mathbf{h}_i$ is the i-th attributes of node v. A bi-directional Long Short-Term Memory (Bi-LSTM) [68] is used to fuse the embeddings learned by multiple attribute encoder $\mathcal{FC}$. Neighbor aggregation aims to aggregate the nodes with same type by using a Bi-LSTM to capture the position information:

$$f_2^t(v) = \frac{\sum_{v'\in N_t(v)}[\overrightarrow{LSTM}\{f_1(v')\} \oplus \overleftarrow{LSTM}\{f_1(v')\}]}{|N_t(v)|}, \tag{16}$$

where $N_t(v)$ is the first-order neighbors of node v with type t. Type aggregation uses an attention mechanism to mix the embeddings of different types and produces the final node embeddings.

$$\mathbf{h}_v = \alpha^{v,v} f_1(v) + \sum_{t\in O_v} \alpha^{v,t} f_2^t(v). \tag{17}$$

where $\mathbf{h}_v$ is the final embedding of node v. O_v denotes the set of node types. Finally, a heterogeneous skip-gram loss is used as the unsupervised graph context loss to update the

node embeddings. Through the three aggregation methods, HetGNN can preserve the heterogeneity of both graph structures and node attributes.

Other unsupervised methods either capture the heterogeneity of node attributes or the heterogeneity of graph structures. HNE [69] is proposed to learn embeddings for the cross-model data in HG, but it ignores the various types of links. SHNE [70] focuses on capturing the semantic information of nodes by designing a deep semantic encoder with gated recurrent units (GRU) [71]. Although it uses heterogeneous skip-gram to preserve the heterogeneity of graph, SHNE is designed specifically for text data. Cen *et al.* propose GATNE [72], which aims to learn node embeddings in multiplex graph, i.e., a heterogeneous graph with different types of edges. Compared with HetGNN, GATNE pays more attention to distinguishing different link relationships between the node pairs.

3.2.2 Semi-supervised HGNNs

Different from unsupervised HGNNs, semi-supervised HGNNs aim to learn task-specific node embeddings in an end-to-end manner. For this reason, they prefer to use attention mechanism to capture the most relevant structural and attribute information to the task.

Wang *et al.* [15] propose heterogeneous graph attention network (HAN), which uses a hierarchical attention mechanism to capture both node and semantic importance. The architecture of HAN is shown in Fig. 5.

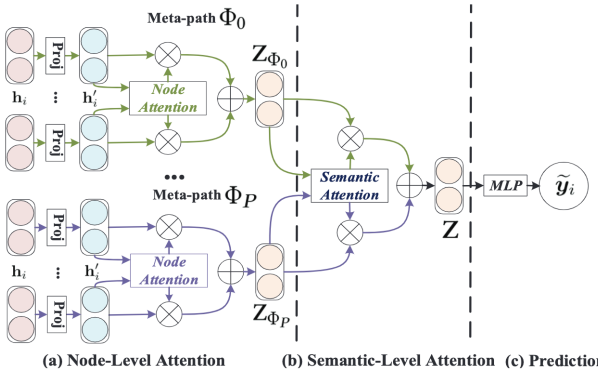


Fig. 5: The architecture of HAN [15]. The whole model can be divided into three parts: Node-Level Attention aims to learn the importance of neighbors' features. Semantic-Level Attention aims to learn the importance of different meta-paths. Prediction layer utilizes the labeled nodes to update the node embeddings.

It consists of three parts: node-level attention, semantic-level attention and prediction. Node-level attention aims to utilize self-attention mechanism [73] to learn the importance of neighbors in a certain meta-path:

$$\alpha_{ij}^{m} = \frac{\exp(\sigma(\mathbf{a}_m^{\mathrm{T}} \cdot [\mathbf{h}_i^{'} \| \mathbf{h}_j^{'}]))}{\sum_{k \in \mathcal{N}_i^m} \exp(\sigma(\mathbf{a}_m^{\mathrm{T}} \cdot [\mathbf{h}_i^{'} \| \mathbf{h}_k^{'}]))}, \tag{18}$$

where $\mathcal{N}_i^m$ is the neighbors of node i in meta-path m, α_{ij}^m is the weight of node j to node i under meta-path m. The node-level aggregation is defined as:

$$\mathbf{h}_i^m = \sigma\left(\sum_{j \in \mathcal{N}_i^m} \alpha_{ij}^m \cdot \mathbf{h}_j\right), \tag{19}$$

where $\mathbf{h}_i^m$ denotes the learned embedding of node i based on meta-path m. Because different meta-paths capture different semantic information of HG, a semantic-level attention mechanism is designed to calculated the importance of meta-paths. Given a set of meta-paths $\{m_0, m_1, \cdots, m_P\}$, after feeding node features into node-level attention, it has P semantic-specific node embeddings $\{\mathbf{H}_{m_0}, \mathbf{H}_{m_1}, \cdots, \mathbf{H}_{m_P}\}$. To effectively aggregate different semantic embeddings, HAN designs a semantic-level attention mechanism:

$$w_{m_i} = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \mathbf{q}^{\mathrm{T}} \cdot \tanh(\mathbf{W} \cdot \mathbf{h}_i^m + \mathbf{b}), \tag{20}$$

where $\mathbf{W} \in \mathbb{R}^{d' \times d}$ and $\mathbf{b} \in \mathbb{R}^{d' \times 1}$ denote the weight matrix and bias of the MLP, respectively. $\mathbf{q} \in \mathbb{R}^{d' \times 1}$ is the semantic-level attention vector. In order to prevent the node embeddings from being too large, HAN uses the softmax function to normalize w_{m_i}. Hence, the semantic-level aggregation is defined as:

$$\mathbf{H} = \sum_{i=1}^{P} \beta_{m_i} \cdot \mathbf{H}_{m_i}, \tag{21}$$

where β_{m_i} denotes the normalized w_{m_i}, which represents the semantic importance. $\mathbf{H} \in \mathbb{R}^{N \times d}$ denotes the final node embeddings. Finally, a task-specific layer is used to fine-tune the node embeddings with a small number of labels and the embeddings $\mathbf{H}$ can be used in the downstream tasks, such as node clustering and link prediction. HAN is the first to extend GNN to the heterogeneous graph and design a hierarchical attention mechanism, which can capture both structural and semantic information.

Then a series of attention-based HGNNs was proposed [74], [75], [76], [77]. MAGNN [74] designs intra-metapath aggregation and inter-metapath aggregation. The former samples some meta-path instances surrounding the target node and uses an attention layer to learn the importance of different instances, and the latter aims to learn the importance of different meta-paths. HetSANN [75] and HGT [76] treat one type of node as query to calculate the importance of other types of nodes around it, through which the method can not only capture the interactions among different types of nodes, but also assign different weights to neighbors during aggregation. [77] uses meta-paths as virtual edges to enhance the performance of graph attention operator.

In addition, there are some HGNNs that focus on other issues. NSHE [78] proposes to incorporate network schema, instead of meta-path, in aggregating neighborhood information. GTN [79] aims to automatically identify the useful meta-paths and high-order links in the process of learning node embeddings. RSHN [80] uses both original node graph and coarsened line graph to design a relation-structure aware HGNN. RGCN [81] uses multiple weight matrices to project the node embeddings into different relation spaces, thus capturing the heterogeneity of the graph.

3.2.3 Summary

As we can see that there are two ways to solve the heterogeneity of attributes: one is to use different encoders or type-specific transformation matrices to map the different attributes into a same space, such as [16], [69]. Another is to treat meta-path as a special edge to connect the nodes with same type, such as [15], [74]. Compared with shallow models, HGNNs have an obvious advantage that they have the ability of inductive learning, i.e., learning embeddings for the out-of-sample nodes [24]. Besides, HGNNs need smaller memory space because they only need to store model parameters. These two reasons are important for the real-world applications. However, they still suffer from the huge time costing in inferencing and retraining.

3.3 Application-oriented HG Embedding

Heterogeneous graph embedding also has been closely combined with some specific applications, where the aforementioned information, e.g., attributes, is not sufficient for specific applications. Under such settings, one usually needs to carefully consider two factors: the first is how to construct a HG for a specific application, and the second is what information, i.e., domain knowledge, should be incorporated into heterogeneous graph embedding, so as to finally benefit the application. In this section, we discuss three common types of applications: recommendation, identification and proximity search.

3.3.1 Recommendation

In recommendation system, the interaction among user and item can be naturally modeled as a HG with two types of nodes. Therefore, recommendation is a typical scenario that widely uses HG information [13]. Besides, other types of information, such as the social relationships, can also be easily introduced in HG [82], applying heterogeneous graph embedding to recommendation application is an important research field.

Early works recommend item to a user mainly based on meta-path aware similarity between user and item, such as HeteLearn [83] and SemRec [82]. With the development of embedding technology, matrix factorization [84], [85], [86], random walk [2] and advanced neural networks [3], [87], [88], [89], [20], [19] are proposed to learn embeddings of user and item, so as to capture the complex interactions.

HERec [2] aims to learn the embeddings of users and items under different meta-paths and fuses them for recommendation. It first finds the co-occurrence of users and items based on the meta-path guided random walks on user-item HG. Then it uses node2vec [90] to learn preliminary embeddings from the co-occurrence sequences of users and items. Because the embeddings under different meta-paths contain different semantic information, for better recommendation performance, HERec designs a fusion function to unify the multiple embeddings:

$$g(\mathbf{h}_u^m) = \frac{1}{|P|} \sum_{m=1}^{M} (\mathbf{W}^m \mathbf{h}_u^m + \mathbf{b}^m), \tag{22}$$

where $\mathbf{h}_u^m$ is the embedding of user node u in meta-path m. M denotes the set of meta-paths. The fusion of item embeddings is similar to users. Finally, a prediction layer is used to predict the items that users prefer. HERec optimizes the graph embedding and recommendation objective jointly.

Apart from random walk, some methods try to use matrix factorization to learn user and item embeddings. HeteRec [86] considers the implicit user feedback in HG. HeteroMF [84] designs a heterogenous matrix factorization technique to consider the context dependence of different types of nodes. FMG [85] incorporates meta-graphs into embedding technology, which can capture some special patterns between users and items.

Previous methods mainly use shallow models to learn the embeddings of users and items, where the ability of express nonlinear interaction between them is limited. Therefore, some neural network-based methods are proposed. One of the most important techniques is attention mechanism, which aims to find the important users and items in HG based recommendation. MCRec [3] designs a neural co-attention mechanism to capture the relationship between user, item and meta-path. Specifically, it uses the users and items to find the important meta-paths. Meanwhile, the important meta-paths are used to find the important users and items in recommendation. Through this mutual selective attention mechanism, MCRec can not only learn embeddings of users, items and meta-paths, but also capture the complex interactions among them. NeuACF [87] and HueRec [89] first calculate multiple meta-path-based commuting matrices, where each row represents the user-user similarity or item-item similarity. Then an attention mechanism is designed to learn the importance of different meta-path-based commuting matrices, so as to capture different semantic information.

Another type of important techniques is graph neural networks. PGCN [88] converts the user-item interaction sequences into item-item graph, user-item graph and user-sequence graph. Then it designs a HGNN to propagate user and item information in the three graphs, so as to capture the collaborative filtering signals. MEIRec [19] focuses on the problem of intent recommendation in E-commerce, which aims to automatically recommend user intent according to user historical behaviors. It constructs a user-item-query heterogeneous graph and designs a meta-path-guided HGNN to learn the embedding of users, items and queries, which can capture the intent of users. GNewsRec [91] and GNUD [5] are designed for news recommendation. They consider both the content information of news and the collaborative information between users and news. [92] employs graph convolutional network on heterogeneous graphs for basket recommendation.

3.3.2 Identification

Identification is to find the most likely nodes according to the given conditions on HG. For example, finding potential authors of a given paper or identifying users in cross-platform. Currently, two representative identification applications, author identification [44], [93], [94] and user identification [95], [96], [97], have been studied based on heterogeneous graph embedding.

Author identification aims to find the potential authors for an anonymous paper in the academic network. Camel [93] aims to consider both the content information, e.g.,

the text of papers, and context information, e.g., the co-occurrence of paper and author. For content information, it designs a content encoder to learn embeddings from the abstract of paper and a metric-based loss function is used to learn the pair-wise relations between authors and papers:

$$\mathcal{L}_{Metric} = \xi + \|f(\mathbf{h}_v) - \mathbf{h}_u\|^2 - \|f(\mathbf{h}_v) - \mathbf{h}_{u'}\|^2, \quad (23)$$

where ξ is the margin, $f(\cdot)$ represents the content encoder and $\mathbf{h}_v$, $\mathbf{h}_u$ and $\mathbf{h}_{u'}$ denote the attributes of paper, positive author and negative author, respectively. For context information, a meta-path guided walk integrative learning module (MWIL) is proposed to preserve the graph structures:

$$\mathcal{L}_{MWIL} = -\log\sigma[f(\mathbf{h}_v)\cdot\mathbf{h}_u] - \log\sigma[-f(\mathbf{h}_v)\cdot\mathbf{h}_{u'}]. \quad (24)$$

It is worth noting that $\mathcal{L}_{MWIL}$ is a special skip-gram technique, which aims to preserve the proximity of positive author u of paper v within a walk length. Through optimizing $\mathcal{L}_{Metric}$ and $\mathcal{L}_{MWIL}$ jointly, Camel considers both the heterogeneous graph structures and the pair-wise relation of author-paper. Similar to the idea of Camel, PAHNE [44] considers the pair-wise relations and TaPEm [94] maximizes the proximity between the paper-author pair and the context path around them.

Compared with author identification, user identification does not contain the pair-wise relation, i.e., user and paper. Therefore, it focuses on learning discriminating user embeddings with weak supervision information so that the target users can be identified more easily. Player2vec [95], AHIN2vec [96] and Vendor2vec [97] are the principal methods. They can be summarized as a general framework: first, some advanced neural networks, e.g., convolutional neural network (CNN) or recurrent neural network (RNN), are used to learn preliminary node embeddings from the raw features. Then the preliminary node embeddings will be propagated on the graphs, constructed by different meta-paths, to utilize the neighborhood information. Finally, a semi-supervised loss function is used to make the node embeddings contain application-specific information. Under the guidance of partially labeled nodes, the node embeddings can distinguish special users from the ordinary users in the graph, which can be used for user identification.

3.3.3 Proximity Search

Given a target node in HG, the proximity search, as shown in Fig. 6, is to find the nodes that are closest to the target node by using structural and semantic information of HG. Some earlier studies have deal with this problem in homogeneous graphs, for example, web search [98]. Recently, some methods try to utilize HG in proximity search [34], [99]. However, these methods only use some statistical information, e.g., the number of connected meta-paths, to measure the similarity of two nodes in HG, which lack flexibility. With the development of deep learning, some embedding methods are proposed.

Prox [100] uses heterogeneous graph embedding in semantic proximity search. Given a set of training tuples $\{q_i, v_i, u_i\}$, where q_i is the query node and in each query the similarity $S(q_i, v_i)$ between node v_i and q_i is larger than $S(q_i, u_i)$. It firstly samples some heterogeneous sequences for each node in the training tuples and feed them into a LSTM ito learn node embeddings. A ranking-based loss function is used to use the implicit supervision information:

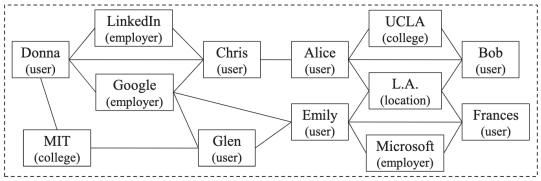


Fig. 6: An example of semantic proximity search [45], which gives a query object (e.g., *Alice*) and requires the method rank other objects according to the semantic relation (e.g., "who are likely to be her *schoolmates*?").

$$L(S(q_i, v_i), S(q_i, u_i)) = -\log\sigma(S(q_i, v_i) - S(q_i, u_i)). \quad (25)$$

Minimizing the function indicates that the similarity between v_i and q_i should be larger than that between u_i and q_i. Different from previous methods that use manually calculated similarities, Prox uses heterogeneous graph embedding to avoid the feature engineering for semantic proximity search, which is an efficient and effective approach.

Then a series of methods are proposed. IPE [45] considers the interactions among different meta-path instances and propose an interactive-paths structure to improve the performance of heterogeneous graph embedding. SPE [101] proposes a subgraph-augmented heterogeneous graph embedding method, which uses a stacked autoencoder to learn the subgraph embedding so as to enhance the effect of semantic proximity search. D2AGE [102] explores the DAG structure for better measuring the similarity between two nodes and designs a DAG-LSTM to learn node embeddings.

3.3.4 Summary

Incorporating heterogeneous graph embedding into specific applications usually need to consider the domain knowledge. For example, in recommendation, meta-path "user-item-user" can be used to capture the user-based collaborative filtering, while "item-user-item" represents the item-based collaborative filtering; in proximity search, methods use meta-paths to capture the semantic relationships between nodes, thus enhancing the performance. Therefore, utilizing HG to capture the application-specific domain knowledge is essential for application-oriented heterogeneous graph embedding.

3.4 Dynamic HG Embedding

In the beginning of Section 3, we mention that previous HG surveys [32], [33] focus on summarizing the static methods, while the dynamic methods are largely ignored. Since the real-world graphs are constantly changing over time, to fill this gap, in this section, we summary the dynamic heterogeneous graph embedding methods. Specifically, they can be divided into two categories: incremental update and retrained update methods. The former learns the embedding of new node in the next timestamp by utilize existing node embeddings, while the latter will retrain the models in each timestamp. Both of them have its own pros and cons, and will be discussed in the end.

3.4.1 Incremental HG Embedding

DyHNE [42] is an incremental update method based on the theory of matrix perturbation, which learns node embeddings while considering both the heterogeneity and evolution of HG. To ensure the effectiveness, DyHNE preserves the meta-path based first- and second-order proximities. The first-order proximity requires two nodes connected by meta-path m to have similar embeddings. And the second-order proximity indicates that the node embedding should be close to the weighted sum of its neighbor embeddings. Specifically, the first- and second-order proximities can be uniformly rewritten as:

$$\mathcal{L} = \mathrm{tr}(\mathbf{H}^\top(\mathbf{L} + \gamma\mathbf{T})\mathbf{H}), \tag{26}$$

where γ is a hyperparameter. $\mathbf{W} = \sum_{m\in M}\theta_m\mathbf{W}^m$ and $\mathbf{D} = \sum_{m\in M}\theta_m\mathbf{D}^m$ are the fusion of different meta-paths, which lead to $\mathbf{L} = \mathbf{D} - \mathbf{W}$ and $\mathbf{T} = (\mathbf{I} - \mathbf{W})^\top(\mathbf{I} - \mathbf{W})$. The minimization of $\mathcal{L}$ can be solved by the eigenvalue decomposition:

$$(\mathbf{L} + \gamma\mathbf{T})\mathbf{H} = \mathbf{D}\Lambda\mathbf{H}, \tag{27}$$

where $\Lambda = diag(\lambda_1, \lambda_2 \cdots \lambda_N)$ is the eigenvalue matrix. To model the evolution of HG, DyHNE uses the perturbation of meta-path augmented adjacency matrices to capture changes of graph. At a new timestamp, the matrix becomes:

$$\begin{aligned}&(\mathbf{L} + \Delta\mathbf{L} + \gamma\mathbf{T} + \gamma\Delta\mathbf{T})(\mathbf{h}_i + \Delta\mathbf{h}_i)\\ &= (\lambda_i + \Delta\lambda_i)(\Lambda + \Delta\Lambda)(\mathbf{h}_i + \Delta\mathbf{h}_i).\end{aligned} \tag{28}$$

where Δ denote the perturbation term. $\Delta\mathbf{h}$ and $\Delta\lambda$ are the changes of the eigenvectors and eigenvalues. Hence, the incremental update of node i is how to calculate the changes of the i-th eigen-pair $(\Delta\mathbf{h}_i, \Delta\lambda_i)$. With some approximations, DyHNE can directly update the node embeddings without retraining the whole model. Generally speaking, DyHNE preserves both the structural and semantic information of HG and uses the perturbation of matrix to capture the evolution of HG over time, which is an effective and efficient method. [103], [104] also adopt the idea of incremental update. Change2vec [103] proposes a dynamic version of metapath2vec. MetaDynaMix [104] uses the incremental update on the matrix factorization of HG.

3.4.2 Retrained HG Embedding

Retrained update methods first use GNNs to learn node or edge embeddings in each timestamp and then design some advanced neural network, e.g., RNN or attention mechanism, to capture the temporal information of HG.

DyHATR [105] aims to capture the temporal information through the changes of nodes embeddings in different timestamps. To this end, as shown in Fig. 7, it first designs a hierarchical attention mechanism (HAT), which contains a node- and edge-level attention, to learn node embeddings by fusing the attributes of neighbors. The node-level attention is defined as:

$$\alpha_{i,j}^{rt} = \frac{\exp(\sigma(\mathbf{a}_r^{\mathrm{T}} \cdot [\mathbf{M}_r \cdot \mathbf{h}_i || \mathbf{M}_r \cdot \mathbf{h}_j]))}{\sum_{k\in\mathcal{N}_i^{rt}}\exp(\sigma(\mathbf{a}_r^{\mathrm{T}} \cdot [\mathbf{M}_r \cdot \mathbf{h}_i || \mathbf{M}_r \cdot \mathbf{h}_j]))}, \tag{29}$$

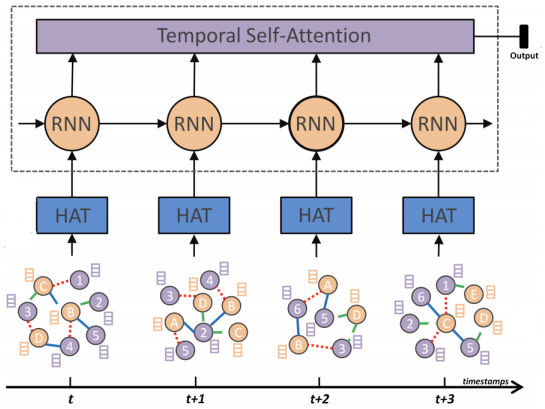


Fig. 7: The architecture of DyHATR [105]. It consists of two parts: first, a hierarchical attention mechanism is designed to learn node embeddings by fusing the attributes of neighbors. Then, a RNN with self-attention mechanism is used to capture the temporal information.

where $\mathcal{N}_i^{rt}$ represents the neighbors of node i in edge type r and timestamp t, and $\mathbf{a}_r$ is the attention vector. And the edge-level attention is:

$$\beta_i^{rt} = \frac{\exp(\mathbf{q}^{\mathrm{T}} \cdot \sigma(\mathbf{W} \cdot \mathbf{h}_i^{rt} + \mathbf{b}))}{\sum_{r\in R}\exp(\mathbf{q}^{\mathrm{T}} \cdot \sigma(\mathbf{W} \cdot \mathbf{h}_i^{rt} + \mathbf{b}))}, \tag{30}$$

where $\mathbf{q}^{\mathrm{T}}$ is the attention vector in edge-level attention. Through the node- and edge-level attention, DyHATR can learn the node embeddings under different timestamps. In order to capture the temporal information hidden in the changes of node embeddings, the node embeddings are fed into a RNN in the order of timestamps. Coincidentally, DyHAN [43] also designs a hierarchical attention mechanism to learn the importance of nodes and timestamps, respectively.

3.4.3 Summary

It can be seen that the incremental update methods are efficient, but they can only capture the short-term temporal information (i.e., the last timestamp) [105]. Besides, incremental update methods focus on utilizing shallow model, which lacks the non-linear expressive power. On the contrary, the retrained update methods employ neural networks to capture the long-term temporal information. However, they suffer from the high computational cost. Therefore, how to combine the advantages of these two kinds of models is an important problem. In addition, there are some meaningful problems to consider, e.g., how to eliminate the cumulative errors in incremental update methods.

3.5 Miscellanea

In the previous section, we introduce the major applications in HG embedding. There are also some other methods that do not belong to the existing categories.

The first is incorporating HG into natural language processing (NLP). Due to the multiple elements in the corpus, e.g., words, entities, sentences or paragraphs, many NLP tasks can be modeled by HG naturally. Graph-to-sequence (Graph2Seq) learning is an important topic in NLP, which aims to transform the graph-structured embeddings to word

sequences for text generation [106], [107]. AMR-to-text generation is a typical Graph2Seq task. It generates text from the Abstract Meaning Representation (AMR) graph, where nodes represent the semantic concepts in the text and edges denote the relations between concepts. In order to learn useful information from the AMR graph, Yao *et al.* [108] treat AMR graph as a HG and design a HG encoder to learn the semantic information among the concepts. Besides, Hu *et al.* [4] propose HGAT for short text classification, which treats the topic, entities and documents as a HG and designs a hierachical attention to learn the similarity among short texts. GNewsRec [91] and GNUD [5] use HG to model the collaborative between news and users in news recommendation task. [109] incorporates HG into topic model for aspect mining. [110] uses HG in fake new detection.

Similar to NLP, multi-modal data can also be modeled by HG due to the various data forms, e.g., text, images or videos. The potential connections among multi-model data can be modeled by HG easily. Therefore, some methods try to use HG embedding to capture the potential dependencies and connections. For example, Community Question Answering (CQA) aims to recommend the suitable answers for each question. Because the answers and questions may contain text and pictures, [111], [112] treats the answers and question as a HG to capture the potential connections, thus making the state-of-the-art performance.

Besides, graph embedding in hyperbolic space has received widespread attention [113], [114], [115]. Because whether Euclidean spaces are the optimal isometric spaces is still an unsolved problem, exploring HG embedding in the hyperbolic spaces is a meaningful research direction. [62] shows that hyperbolic spaces can capture the hierarchical and power-law structure of the HG, which provides a theoretical guarantee for the future work to some extent.

Moreover, HG embedding are also widely used to model many other tasks, such as entity set expansion [116], basket recommendation [117], event categorization [118] and social network [119].

4 Technique Summary

In the previous section, we category the HG embedding methods based on different problem setting. In this section, from the technical perspective, we summarize the widely used techniques (or models) in HG embedding, which can be generally divided into two categories: shallow model and deep model.

4.1 Shallow Model

Early HG embedding methods focus on employing shallow model. They first initialize the node embeddings randomly, and then learn the node embeddings through optimizing some well-designed objective functions. We divide the shallow model into two categories: random walk-based and decomposition-based.

Random walk-based. In homogeneous graph, random walk, which generates some node sequences in a graph, is usually used to capture the local structure of a graph [90]. While in HG, the node sequence should contain not only the structural information, but also the semantic information. Therefore, a series of semantic-aware random walk techniques are proposed [57], [8], [59], [60], [61], [62], [2]. For example, metapath2vec [8] uses meta-path-guided random walk to capture the semantic information of two nodes, e.g., the co-author relationship in academic graph. Spacey [59] and metagraph2vec [41] design metagraph-guided random walks, which preserve a more complex similarity between two nodes.

Decomposition-based. Decomposition-based techniques aim to decompose HG into several sub-graphs and preserve the proximity of nodes in each sub-graph [17], [50], [52], [53], [55], [56], [66]. PME [17] decomposes the HG into some bipartite graphs according to the types of links and projects each bipartite graph into a relation-specific semantic space. PTE [56] divides the documents into word-word graph, word-document graph and word-label graph. Then it uses LINE [39] to learn the shared node embeddings for each sub-graph. HEBE [66] samples a series of subgraphs from a HG and preserves the proximity between the center node and its subgraph.

4.2 Deep Model

Deep model aims to use advanced neural networks to learn embedding from the node attributes or the interactions among nodes, which can be roughly divided into three categories: message passing-based, encoder-decoder-based and adversarial-based.

Message passing-based. The idea of message passing is to send the node embedding to its neighbors, which is always used in GNNs. The key component of message passing-based techniques is to design a suitable aggregation function, which can capture the semantic information of HG [15], [74], [75], [16], [72], [78], [79], [80], [81]. HAN [15] designs a hierarchical attention mechanism to learn the importance of different nodes and meta-paths, which captures both structural information and semantic information of HG. HetGNN [16] uses bi-LSTM to aggregate the embedding of neighbors so as to learn the deep interactions among heterogeneous nodes. GTN [79] designs an aggregation function, which can find the suitable meta-paths automatically during the process of message passing.

Encoder-decoder-based. Encoder-decoder-based techniques aim to employ some neural networks as encoder to learn embedding from node attributes and design a decoder to preserve some properties of the graphs [65], [69], [70], [44], [93], [94]. For example, HNE [69] focuses on multi-modal HG. It uses CNN and autoencoder to learn embedding from images and texts, respectively. Then it uses the embedding to predict whether there is a link between the images and texts. Camel [93] uses GRU as encoder to learn paper embedding from the abstracts. A skip-gram objective function is used to preserve the local structures of the graphs. DHNE [65] uses autoencoder to learn embedding for the nodes in a hyperedge. Then it designs a binary classification loss to preserve the indecomposability of the hyper-graph.

Adversarial-based. Adversarial-based techniques utilize the game between generator and discriminator to learn robust node embedding. In homogeneous graph, the adversarial-based techniques only consider the structural

TABLE 2: Typical heterogeneous graph embedding methods.

Method	Inductive	Label	Information	Task	Technique	Characteristic
mp2vec [8]			Strcuture	Embedding	Random walk (Shallow model)	• Easy to parallelize • Two-stage training • High memory cost Complexity: $\mathcal{O}(\tau \cdot l \cdot k \cdot n_s \cdot d \cdot \lvert\mathcal{V}\rvert)$
Spacey [59]						
JUST [60]						
BHIN2vec [61]						
HHNE [62]						
mg2vec [41]						
HeRec [2]		√	Strcuture+Task	Recommendation		
PME [17]			Strcuture	Embedding	Decomposition (Shallow model)	• Easy to parallelize • Two-stage training • High memory cost Complexity: $\mathcal{O}(\lvert\mathcal{E}\rvert \cdot d)$
EOE [50]						
HEER [53]						
MNE [57]						
PTE [17]						
RHINE [63]						
HAN [15]	√	√	Structure+Attribute		Message passing (Deep model)	• End-to-End training • Encoding structures and attributes • Semantic fusion • High training cost Complexity: $\mathcal{O}(\lvert\mathcal{V}\rvert \cdot d_1 + \lvert\mathcal{R}\rvert \cdot d_2)$
MAGNN [74]	√	√				
HetSANN [75]	√	√				
HGT [76]	√	√				
HetGNN [16]	√					
GATNE [72]	√					
GTN [79]		√				
RSHN [80]		√				
RGCN [81]	√	√				
IntentGC [20]	√	√	Strcuture +Attribute+Task	Recommendation		
MEIRec [19]	√	√				
GNUD [5]	√	√				
Player2vec [95]	√	√		Identification		
AHIN2vec [96]	√	√				
Vendor2vec [97]	√	√				
HIN2vec [9]			Strcuture	Embedding	Encoder-decoder (Deep model)	• End-to-End training • Flexible goal-orientation Complexity: $\mathcal{O}(\lvert\mathcal{V}\rvert \cdot d_1 + \lvert\mathcal{E}\rvert \cdot d_2)$
DHNE [65]						
HNE [69]	√	√	Structure+Attribute			
SHNE [70]		√				
NSHE [78]						
PAHNE [44]		√	Strcuture +Attribute+Task	Identification		
Camel [93]		√				
TaPEm [94]		√				
HeGAN [18]			Strcuture	Embedding	Adversarial (Deep model)	• Robustness • High complexity Complexity: $\mathcal{O}(\lvert\mathcal{V}\rvert \cdot \lvert\mathcal{R}\rvert \cdot n_s \cdot d)$
MV-ACM [120]						
Rad-HGC [24]		√	Strcuture+Task	Malware detection		

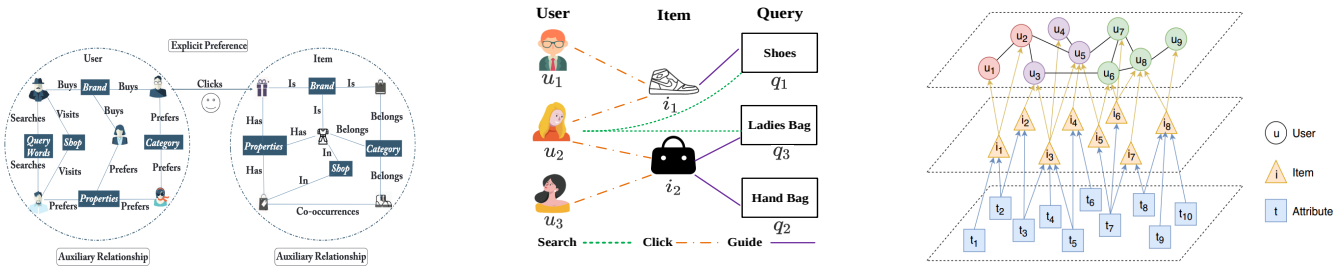
information, for example, GraphGAN [122] uses Breadth First Search when generating virtual nodes. In a HG, the discriminator and generator are designed to be relation-aware, which captures the rich semantics on HGs. HeGAN [3] is the first to use GAN in HG embedding. It incorporates multiple relations into generator and discriminator, so that the heterogeneity of a given graph can be considered. MV-ACM [120] uses GAN to generate the complementary views by computing the similarity of nodes in different views.

4.3 Review

In Table 2, we categorize the typical HG embedding methods through different perspectives. The first two columns indicate whether the method has inductive capability and whether it needs labels for training. We can see that most message passing-based methods have the inductive capability because they can update the node embeddings by aggregating neighborhood information. But they need additional labels to guide the training process.

The middle two columns show the information and task in each method. It can be seen that most deep model-based methods are proposed for HG with attributes or specific application, while the shallow model-based methods are mainly designed for the use of structures. One possible reason is that HG with attributes or specific applications usually needs to introduce additional information or domain knowledge. However, modeling the domain knowledge may be complicated. Deep model provides a more powerful support for this kind of complex modeling, and it helps to make better progress in the complex application scenarios. Meanwhile, the emerging HGNNs can naturally integrate graph structures and attributes, so it is more suitable for the complex scenes and content.

The last two columns summarize the techniques used in HG embedding and their characteristics. Shallow models are easy to parallel. But they are two-stage training, i.e., the embeddings are not relevant to the downstream tasks, and the memory cost is heavy. On the contrary, deep models are end-to-end training and require less memory



(a) E-commerce recommendation HG [20]. (b) Intent recommendation HG [19]. (c) User Profiling HG [121].

Fig. 8: The representative HGs in E-commerce.

space. Besides, message passing-based techniques are good at encoding structures and attributes simultaneously, and integrating different semantic information. Compared with message passing-based techniques, encoder-decoder-based techniques are weak in fusing information due to the lack of messaging mechanism. But they are more flexible to introduce different objective function through different decoders. Adversarial-based methods prefer to utilize the negative samples to enhance the robustness of the embeddings. But the choice of negative samples has a huge influence on the performance, thus leading higher variances [18].

It is worth noting that we also list the complexity of each techniques, where τ is the number of random walks, l is the length of random walk, k is the windows size in skip-gram [58] and n_s is the number of samples.

5 Real-world Deployed Systems

HG embedding is closely related with the real-world applications, as heterogeneous objects and interactions are ubiquitous in many practical systems. Here we focus on summarizing the industrial level applications with HG embedding. For industrial-level applications, we pay more attention to two key components: HG construction with industrial data and graph embedding techniques on the HG.

5.1 E-commerce

E-commerce, such as Taobao[2] and Amazon[3], is the activity of electronic trading of products on online services. Large-scale heterogeneous objects and interactions, such as users, items, and shops, are involved in an e-commerce platform. HG can naturally model such complex data and HG embedding has been applied to various important services and tasks in e-commerce, such as item/intent recommendation, user profiling, and fraudster detection.

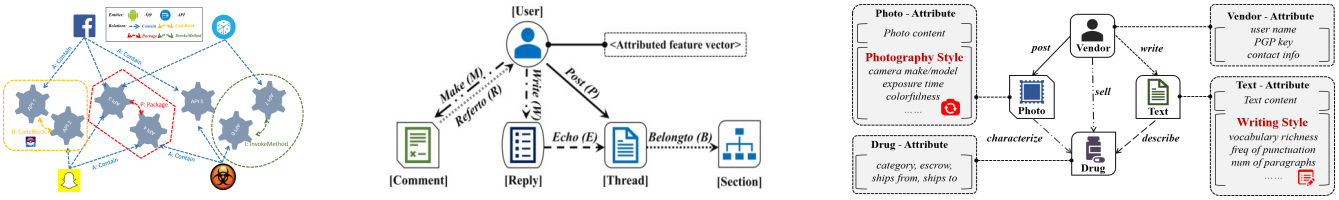
Recommendation is an important service of an e-commerce platform. HG can be used to model the interactions among users, items, and their auxiliary information [82]. As shown in Fig. 8(a), the HG constructed by IntentGC [20] is composed of user part and item part, and each part models the corresponding heterogeneous relationships. IntentGC translates the original HG as a multi-relation graph of users and items and develops a multi-relation graph convolution method to learn node embeddings. GATNE [72] distinguishes the interactions between user and item pairs as multiple types, models this scenario as an attributed multiplex HG and proposes an unified embedding method that captures both attribute and edge information. More recently, to solve the interaction sparsity problem, Xu *et al.* [123] transform the original user-item HG into two homogeneous graphs from the perspective of users and items respectively.

Different from recommending items for users, intent recommendation aims to automatically recommend user intent according to user historical behaviors without any input. Fan *et al.* [19] propose to represent user intent as default queries in search box and transform the intent recommendation problem as recommending the queries. They construct a HG containing three types of nodes (Users, Items and Queries) and their mutual interactions, shown in Fig. 8(b). Then, a meta-path-guided HGNN, called MEIRec, is designed to learn the nodes' embeddings of users and queries through aggregating the neighbors along the given meta-paths in an end-to-end manner.

User profiling plays an increasingly important role in providing personalized services in e-commerce platform. It models the abundant interaction information of users as a HG to enrich the characteristics of users. Chen *et al.* [121] construct three kinds of objects (i.e., users, items and attributes) as a HG, shown in Fig. 8(c), and propose a hierarchical heterogeneous GAT to predict the traits of users (e.g., gender and age) by aggregating each layer of objects' embeddings. Apart from trait prediction, Zheng *et al.* [124] exploit HG to model the interactions between PID and MID with item ID in the e-commerce user alignment task. Then a Heterogeneous Embedding Propagation (HEP) model, encoding the interaction and edge features into node embeddings, is proposed to predict whether PID and MID across different devices refer to the same person.

With the development of e-commerce, there are many fraudsters in e-commerce system, who profit from transactions by illegal means. Due to the heterogeneity of fraudsters behavior patterns, some works try to detect these malicious accounts through HG embedding methods. Liu *et al.* [21] consider both the device and activity of fraudsters, and propose a HGNN, called GEM, which simultaneously models the topology of the heterogeneous account-device graph and the characteristics of accounts activities in the local structure. Moreover, to enrich the embeddings of users, Hu *et al.* [6] treat the users, merchants, devices in credit payment service as different types of nodes and their

2. www.taobao.com
3. www.amazon.com



(a) Malware detection [7]. (b) Key player identification [95]. (c) Drug trafficker identification [97].

Fig. 9: The representative HGs in cybersecurity applications.

interactions as edges in a HG, and propose a meta-path-based HG embedding method, called HACUD, to classify the cash-out user. Li *et al.* [125] treat the users and items as nodes in a bipartite graph and associate the reviews as edge features to detect the spam reviews on Xianyu App.

5.2 Cybersecurity

Security has been one of the biggest threaten for social development, and it causes countless loss of property and lives. As multiple heterogeneous entities and complex structure are usually involved in security system, recently researchers pay more attention to use HG embedding methods to detect outliers in a wide range of security areas, such as malware detection, key player identification in underground forum, drug trafficker identification.

With the broad scale proliferation of increasingly interconnected devices, malware (e.g., trojans, ransomware, scamware) that deliberately fulfills the harmful intent to device users has become a major threat to compromise the security in cyberspace [126]. In particular, the explosive growth and increasing sophistication of Android malware call for new defensive techniques that are capable of protecting mobile users against novel threats [127].

To combat the evolving Android malware attacks, HG-based methods have been proposed and applied in anti-malware industry. As shown in Fig. 9(a), HinDroid [7] was first proposed to construct a HG to model the complex relations among application programming interface (APIs) and Android applications (apps), based on which meta-paths are used to formulate the relatedness among apps and multi-kernel learning algorithm is proposed to build the classification model for malware detection. Besides modeling apps and APIs, Fan *et al.* [22] model more types of entities involved in malware into a HG, such as, file, archive and machine, and a metagraph based embedding method is designed to encode high-level semantic similarities between files. After these methods, a series of HG embedding methods are proposed for dynamic malware detection [23], adversarial attack and defense in malware [24], unknown malware detection [128] and cyber threat intelligence [129].

Besides android malware detection, HG embedding methods also play an important role in detecting targeted objects in other security areas which have multiple types of entities and relations available. Zhang *et al.* [95] extract multiple relations from the underground forum data and construct an attributed HG (AHG) for key player identification, shown in Fig. 9(b). By treating the relatedness over users depicted by each meta-path as one view, a multi-view GCN is proposed to identify the key player. As illustrated in Fig. 9(c), Zhang *et al.* [97] leverage AHG to depict vendors, drugs, texts, photos and their associated attributes in darknet markets for drug trafficker identification. Then an attribute-aware embedding method, named *Vendor2Vec*, consisting of attribute-aware meta-path random walk and skip-gram technique, is proposed to predict whether a given pair of vendors are the same individual or not.

5.3 Others

With the development of biological medicine, medical informatics has received considerable attentions, especially, mining Electronic Health Records (EHR) for improving quality of disease diagnosis [25]. Previous work on medical HG utilizes HeteSim [99] to analyze the similarities between objects [130]. Recently, Hosseini *et al.* [26] treat diagnostic and treatments as nodes and edges extracted from raw text in a HG, and propose a meta-path-guided HG embedding method to rank each patient's potential diagnosis.

Besides, HG embedding is also applied in real-time event prediction on ride-hailing platform, such as Uber[4] and DiDi[5]. Luo *et al.* [131] construct HG for each ongoing event, e.g., PreView page and request, to encode the attributes of event and condition information from its surrounding area. A GNN is proposed to learn the impact of historical actions and the surrounding environment on current events, and generate an event embedding to improve the accuracy of the response model. Hong *et al.* [132] propose HetETA to leverage HG to model the spatiotemporal information in time-of-arrival (ETA) estimation task. A multi-component GNN is proposed to model temporal information from different time spans for ETA task.

6 Benchmarks and Open-source Tools

In this section, we summarize the commonly used datasets of HG embedding. Besides, we also introduce some useful resources and open-source tools about HG embedding.

6.1 Benchmark Datasets

High-quality datasets are essential for academic research. Here, we introduce some popular real world HG datasets, which can be divided into three categories: academic networks, business networks and film networks. Detailed statistical information can be seen in the supplemental material, including types, meta-paths and tasks etc.

4. www.uber.com
5. www.didiglobal.com

- **DBLP**[6] This is a network that reflects the relationship between authors and papers. There are four types of nodes: author, paper, term and venue.
- **Aminer**[7] This academic network is similar to DBLP, but with two additional node types: keyword and conference.
- **Yelp**[8] This is a social media network, including five types of nodes: user, business, compliment, city and category.
- **Amazon**[9] This is an E-commercial network, which records the interactive information between users and products, including co-viewing, co-purchasing, etc.
- **IMDB**[10] This is a film rating network, recording the preferences of users on different films. Each film contains its directors, actors and genre.
- **Douban**[11] This network is similar to IMDB, but it contains more user information, such as group and user location.

6.2 Open-source Code

Source code is important for researchers to reproduce the corresponding method. In the supplemental material, we refer to the related papers of the datasets. Besides, we provide some commonly used website about the graph embedding.

- Stanford Network Analysis Project (SNAP). It is a network analysis and graph mining library, which contains different types of networks and multiple network analysis tools. The address is http://snap.stanford.edu/.
- ArnetMiner (AMiner) [133]. In the early days, it was an academic network used for data mining. Now it becomes to a comprehensive academic system that provides a variety of academic resources. The address is https://www.aminer.cn/.
- Open Academic Society (OAS). It is an open and expanding knowledge graph for research and education, contributed by Microsoft Research and AMiner. It publishes Open Academic Graph (OAG), which unifies two billion-scale academic graphs. The address is https://www.openacademic.ai/.
- HG Resources. It is a website focusing on HGs, which collects a series of papers and divides them into different categories, including classficiation, clustering and embedding. Code and datasets of the popular HG methods are also provided. The address is http://shichuan.org/.

6.3 Available Tools

Open-source platforms and toolkits can help researchers build the workflow of graph embedding quickly and easily. There are many toolkits designed for homogeneous graph, e.g., OpenNE[12] and CogDL[13]. However, the toolkits and platforms for HG are rarely mentioned. To bring this gap, we summary the toolkits and platforms that support HG.

- AliGraph. It is an industrial-grade machine learning platform for graph data, supporting the calculation of hundreds of millions of nodes and edges. Besides, it considers the characteristics of real world industrial graph data, i.e., large-scale, heterogeneous, attributed and dynamic, and makes special optimizations. One instance can be found in https://www.aliyun.com/product/bigdata/product.
- Deep Graph Library (DGL). It is an open-source deep learning platform for graph data, which designs its own data structures and implements many popular methods. Specifically, it provides independent APIs for homogeneous graph, heterogeneous graph and knowledge graph. One instance can be found in https://www.dgl.ai/.
- Pytorch Geometric. It is a geometric deep learning extension library for pytorch. Specifically, it focuses on the methods for deep learning on graphs and other irregular structures. Same as DGL, it also has its own data structures and operators. One instance can be found in https://pytorch-geometric.readthedocs.io/en/latest/.
- OpenHINE. It is an open-source toolkit for HG embedding, which implements many popular HG embedding methods with a unified data interface. One instance can be found in https://github.com/BUPT-GAMMA/OpenHINE.

6. http://dblp.uni-trier.de
7. https://www.aminer.cn
8. http://www.yelp.com/dataset challenge/
9. http://jmcauley.ucsd.edu/data/amazon
10. https://grouplens.org/datasets/movielens/100k/
11. http://movie.douban.com/
12. https://github.com/thunlp/OpenNE
13. https://github.com/THUDM/cogdl

7 Challenges and Future Directions

HG embedding has made great progress in recent years, which clearly shows that it is a powerful and promising graph analysis paradigm. In this section, we discuss additional issues/challenges and explore a series of possible future research directions.

7.1 Preserving HG Structures and Properties

The basic success of HG embedding builds on preserving both HG structures and properties. Meta-path [8] and meta-graph [41] are two typical HG structures. However, selecting the most appropriate meta-path is still very challenging in the real world. An improper meta-path will fundamentally hinder the performance of HG embedding method. Whether we can explore other techniques, e.g., motif [36] or network schema [78] to capture HG structure is worth pursuing. Moreover, if we rethink the goal of traditional graph embedding, i.e., replacing the structure information with the distance/similarity in a metric space, a research direction to explore is whether we can design a HG embedding method which can naturally learn such distance/similarity rather than using pre-defined meta-path/meta-graph.

In addition to the HG structures, some properties, which usually provide additional useful information to model HG, have not been fully considered. One typical property is the dynamics of HG. Despite that the incremental learning on dynamic HG is proposed [42], dynamic HG embedding is still facing big challenges. For example, [103] is only proposed with a shallow model, which greatly limits its embedding ability. How can we learn dynamic HG embedding in deep learning framework is worth pursuing. The other property is the uncertainty of HG, i.e., the generation of HG is usually multi-faceted and the node in a HG contains different semantics. Traditionally, learning a vector embedding usually cannot well capture such uncertainty. Gaussian distribution may innately represent the uncertainty property

[134], [135], which is largely ignored by current HG embedding methods. This suggests a huge potential direction for improving HG embedding.

7.2 Deep Graph Learning on HG Data

We have witnessed the great success and large impact of GNNs, where most of the existing GNNs are proposed for homogeneous graph [136], [137]. Recently, HGNNs have attracted considerable attention [15], [16], [74], [72].

One natural question may arise that what is the essential difference between GNNs and HGNNs. More theoretical analysis on HGNNs are seriously lacking. For example, it is well accepted that the GNNs suffer from over-smoothing problem [138], so will heterogeneous GNNs also have such problem? If the answer is yes, what factor causes the over-smoothing problem in HGNNs since they usually contain multiple aggregation strategies [15], [16].

In addition to theoretical analysis, new technique design is also important. One of the most important directions is the self-supervised learning. It uses the pretext tasks to train the neural networks, thus reducing the dependence on manual labels. [139]. Considering the actual demand that label is insufficient, self-supervised learning can greatly benefit the unsupervised and semi-supervised learning, and has shown remarkable performance on homogeneous graph embedding [140], [141], [142], [143]. Therefore, exploring self-supervised learning on HG embedding is expected to further facilitate the development of this area.

Another important direction is the pre-training of HGNNs [144], [145]. Nowadays, HGNNs are designed independently, i.e., the proposed method usually works well for some certain tasks, but the transfer ability across different tasks is ill-considered. When dealing with a new HG or task, we have to train a HG embedding method from scratch, which is time-consuming and requires large amounts of labels. In this situation, if there is a pre-trained HGNN with strong generalization that can be fine-tuned with few labels, the time and label consumption will reduce.

7.3 Making HG embedding reliable

Except from the properties and techniques in HG, we are also concerned about the ethical issues in HG embedding, such as fairness, robustness and interpretability. Considering that most methods are black boxes, making HG embedding reliable is an important future work.

Fair HG embedding. The embeddings learned by methods are sometimes highly related to certain attributes, e.g., age or gender, which may amplify the societal stereotypes in the prediction results [146], [147]. Therefore, learning fair or de-biased embeddings is an important research direction. There are some researches on the fairness of homogeneous graph embedding [146], [148]. However, the fairness of HG is still an unsolved problem, which is an important research direction in the further.

Robust HG embedding. Also, the robustness of HG embedding, especially the adversarial attacking, is always an important problem [149]. Since many real world applications are built based on HG, the robustness of HG embedding becomes an urgent yet unsolved problem. What is the weakness of HG embedding and how to enhance it to improve the robustness need to be further studied.

Explainable HG embedding. Moreover, in some risk aware scenarios, e.g., fraud detection [6] and bio-medicine [25] , the explanation of models or embeddings is important. A significant advantage of HG is that it contains rich semantics, which may provide eminent insight to promote the explanation of heterogeneous GNNs. Besides, the emerging disentangled learning [150], [151], which divides the embedding into different latent spaces to improve the interpretability, can also be considered.

7.4 Technique Deployment in Real-world Applications

Many HG-based applications have stepped into the era of graph embedding. This survey has demonstrated the strong performance of HG embedding methods on E-commerce and cybersecurity. Exploring more capacity of HG embedding on other areas holds great potential in the future. For example, in software engineering area, there are complex relations among test sample, requisition form, and problem form, which can be naturally modeled as HG. Therefore, HG embedding is expected to open up broad prospects for these new areas and become promising analytical tool. Another area is the biological systems, which can also be naturally modeled as a HG. A typical biological system contains many types of objects, e.g., Gene Expression, Chemical, Phenotype, and Microbe. There are also multiple relations between Gene Expression and Phenotype [152]. HG structure has been applied to biological system as an analytical tool, implying that HG embedding is expected to provide more promising results.

In addition, since the complexity of HGNNs are relatively large and the techniques are difficult to parallelize, it is difficult to apply the existing HGNNs to large-scale industrial scenarios. For example, the number of nodes in E-commerce recommendation may reach one billion [20]. Therefore, successful technique deployment in various applications while resolving the scalability and efficiency challenges will be very promising.

7.5 Others

Last but not least, there are also some important future work that cannot be summarized in the previous sections.

Hyperbolic heterogeneous graph embedding. Some recent researches point out that the underlying latent space of graph may be non-Euclidean, but in hyperbolic space [113]. Some attempts have been made towards hyperbolic HG embedding, and the results are rather promising [114], [115], [62]. However, how to design an effective hyperbolic heterogeneous GNNs is still challenging, which can be another research direction.

Heterogeneous graph structure learning. Under the current HG embedding framework, HG is usually constructed beforehand, which is independent on the HG embedding. This may result in that the input HG is not suitable for the final task. HG structure learning can be further integrated with HG embedding, so that they can promote each other.

Connections with knowledge graph. Knowledge graph embedding has great potential on knowledge reasoning [153]. However, knowledge graph embedding and HG

embedding are usually investigated separately. Recently, knowledge graph embedding has been successfully applied to other areas, e.g., recommender system [154], [155]. It is worth studying that how to combine knowledge graph embedding with HG embedding, and incorporate knowledge into HG embedding.

REFERENCES

[1] Y. Sun and J. Han, "Mining heterogeneous information networks: a structural analysis approach," *SIGKDD Explorations*, vol. 14, no. 2, pp. 20–28, 2012.

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

[3] B. Hu, C. Shi, W. X. Zhao, and P. S. Yu, "Leveraging meta-path based context for top-n recommendation with a neural co-attention model," in *KDD*, 2018.

[4] L. Hu, T. Yang, C. Shi, H. Ji, and X. Li, "Heterogeneous graph attention networks for semi-supervised short text classification," in *EMNLP*, 2019.

[5] L. Hu, S. Xu, C. Li, C. Yang, C. Shi, N. Duan, X. Xie, and M. Zhou, "Graph neural news recommendation with unsupervised preference disentanglement," in *ACL*, 2020.

[6] B. Hu, Z. Zhang, C. Shi, J. Zhou, X. Li, and Y. Qi, "Cash-out user detection based on attributed heterogeneous information network with a hierarchical attention mechanism," in *AAAI*, 2019.

[7] 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.

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

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

[10] X. Li, B. Kao, Z. Ren, and D. Yin, "Spectral clustering in heterogeneous information networks," in *AAAI*, 2019.

[11] M. E. Newman, "Modularity and community structure in networks," *Proceedings of the national academy of sciences*, vol. 103, no. 23, pp. 8577–8582, 2006.

[12] B. Weisfeiler and A. A. Lehman, "A reduction of a graph to a canonical form and an algebra arising during this reduction," *Nauchno-Technicheskaya Informatsia*, vol. 2, no. 9, pp. 12–16, 1968.

[13] C. Shi, Y. Li, J. Zhang, Y. Sun, and P. S. Yu, "A survey of heterogeneous information network analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 1, pp. 17–37, 2017.

[14] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 5, pp. 833–852, 2018.

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

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

[17] H. Chen, H. Yin, W. Wang, H. Wang, Q. V. H. Nguyen, and X. Li, "Pme: projected metric embedding on heterogeneous networks for link prediction," in *KDD*, 2018.

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

[19] S. Fan, J. Zhu, X. Han, C. Shi, L. Hu, B. Ma, and Y. Li, "Metapath-guided heterogeneous graph neural network for intent recommendation," in *KDD*, 2019.

[20] J. Zhao, Z. Zhou, Z. Guan, W. Zhao, W. Ning, G. Qiu, and X. He, "Intentgc: a scalable graph convolution framework fusing heterogeneous information for recommendation," in *KDD*, 2019.

[21] Z. Liu, C. Chen, X. Yang, J. Zhou, X. Li, and L. Song, "Heterogeneous graph neural networks for malicious account detection," in *CIKM*, 2018.

[22] Y. Fan, S. Hou, Y. Zhang, Y. Ye, and M. Abdulhayoglu, "Gotcha-sly malware! scorpion a metagraph2vec based malware detection system," in *KDD*, 2018.

[23] Y. Ye, S. Hou, L. Chen, J. Lei, W. Wan, J. Wang, Q. Xiong, and F. Shao, "Out-of-sample node representation learning for heterogeneous graph in real-time android malware detection," in *IJCAI*, 2019.

[24] S. Hou, Y. Fan, Y. Zhang, Y. Ye, J. Lei, W. Wan, J. Wang, Q. Xiong, and F. Shao, "αcyber: Enhancing robustness of android malware detection system against adversarial attacks on heterogeneous graph based model," in *CIKM*, 2019.

[25] Y. Cao, H. Peng, and P. S. Yu, "Multi-information source HIN for medical concept embedding," in *PAKDD*, 2020.

[26] A. Hosseini, T. Chen, W. Wu, Y. Sun, and M. Sarrafzadeh, "Heteromed: Heterogeneous information network for medical diagnosis," in *CIKM*, 2018.

[27] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Network representation learning: A survey," *IEEE transactions on Big Data*, 2018.

[28] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *Knowledge-Based Systems*, vol. 151, 2018.

[29] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 9, 2018.

[30] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[31] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

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

[33] C. Yang, Y. Xiao, Y. Zhang, Y. Sun, and J. Han, "Heterogeneous network representation learning: Survey, benchmark, evaluation, and beyond," *CoRR*, vol. abs/2004.00216, 2020.

[34] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu, "Pathsim: Meta path-based top-k similarity search in heterogeneous information networks," *Proceedings of the VLDB Endowment*, vol. 4, no. 11, pp. 992–1003, 2011.

[35] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, "Network motifs: simple building blocks of complex networks," *Science*, vol. 298, no. 5594, pp. 824–827, 2002.

[36] Z. Huang, Y. Zheng, R. Cheng, Y. Sun, N. Mamoulis, and X. Li, "Meta structure: Computing relevance in large heterogeneous information networks," in *KDD*, 2016.

[37] W. Zhang, Y. Fang, Z. Liu, M. Wu, and X. Zhang, "mg2vec: Learning relationship-preserving heterogeneous graph representations via metagraph embedding," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[38] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *KDD*, 2014.

[39] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *WWW*, 2015.

[40] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding," in *AAAI*, 2017.

[41] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Metagraph2vec: complex semantic path augmented heterogeneous network embedding," in *PAKDD*, 2018.

[42] X. Wang, Y. Lu, C. Shi, R. Wang, P. Cui, and S. Mou, "Dynamic heterogeneous information network embedding with meta-path based proximity," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[43] L. Yang, Z. Xiao, W. Jiang, Y. Wei, Y. Hu, and H. Wang, "Dynamic heterogeneous graph embedding using hierarchical attentions," in *ECIR*, ser. Lecture Notes in Computer Science, vol. 12036, 2020, pp. 425–432.

[44] T. Chen and Y. Sun, "Task-guided and path-augmented heterogeneous network embedding for author identification," in *WSDM*, 2017.

[45] Z. Liu, V. W. Zheng, Z. Zhao, Z. Li, H. Yang, M. Wu, and J. Ying, "Interactive paths embedding for semantic proximity search on heterogeneous graphs," in *KDD*, 2018.

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

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

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

[49] J. L. Suárez, S. García, and F. Herrera, "A tutorial on distance metric learning: Mathematical foundations, algorithms and software," *arXiv preprint arXiv:1812.05944*, 2018.

[50] L. Xu, X. Wei, J. Cao, and P. S. Yu, "Embedding of embedding (eoe): Joint embedding for coupled heterogeneous networks," in *WSDM*, 2017.

[51] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NeurIPS*, 2014.

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

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

[54] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *NeurIPS*, 2013.

[55] R. Matsuno and T. Murata, "MELL: effective embedding method for multiplex networks," in *WWW*, 2018.

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

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

[58] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NeurIPS*, 2013.

[59] 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.

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

[61] S. Lee, C. Park, and H. Yu, "Bhin2vec: Balancing the type of relation in heterogeneous information network," in *CIKM*, 2019.

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

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

[64] S. Helgason, *Differential geometry, Lie groups, and symmetric spaces*. Academic press, 1979.

[65] K. Tu, P. Cui, X. Wang, F. Wang, and W. Zhu, "Structural deep embedding for hyper-networks," in *AAAI*, 2018.

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

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

[68] Z. Huang, W. Xu, and K. Yu, "Bidirectional LSTM-CRF models for sequence tagging," *CoRR*, vol. abs/1508.01991, 2015.

[69] 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.

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

[71] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

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

[73] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, 2017.

[74] X. Fu, J. Zhang, Z. Meng, and I. King, "Magnn: Metapath aggregated graph neural network forheterogeneous graph embedding," *WWW*, 2020.

[75] H. Hong, H. Guo, Y. Lin, X. Yang, Z. Li, and J. Ye, "An attention-based graph neural network for heterogeneous structural learning," *AAAI*, 2020.

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

[77] Y. Fu, Y. Xiong, P. S. Yu, T. Tao, and Y. Zhu, "Metapath enhanced graph attention encoder for hins representation learning," in *BigData*, 2019.

[78] J. Zhao, X. Wang, C. Shi, Z. Liu, and Y. Ye, "Network schema preserving heterogeneous information network embedding," in *IJCAI*, 2020.

[79] S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim, "Graph transformer networks," in *NeurIPS*, 2019.

[80] S. Zhu, C. Zhou, S. Pan, X. Zhu, and B. Wang, "Relation structure-aware heterogeneous graph neural network," in *IEEE International Conference On Data Mining*, 2019.

[81] 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.

[82] C. Shi, Z. Zhang, P. Luo, P. S. Yu, Y. Yue, and B. Wu, "Semantic path based personalized recommendation on weighted heterogeneous information networks," in *CIKM*, 2015.

[83] Z. Jiang, H. Liu, B. Fu, Z. Wu, and T. Zhang, "Recommendation in heterogeneous information networks based on generalized random walk model and bayesian personalized ranking," in *WSDM*, 2018.

[84] M. Jamali and L. V. S. Lakshmanan, "Heteromf: recommendation in heterogeneous information networks using context dependent factor models," in *WWW*, 2013.

[85] H. Zhao, Q. Yao, J. Li, Y. Song, and D. L. Lee, "Meta-graph based recommendation fusion over heterogeneous information networks," in *KDD*, 2017.

[86] X. Yu, X. Ren, Y. Sun, B. Sturt, U. Khandelwal, Q. Gu, B. Norick, and J. Han, "Recommendation in heterogeneous information networks with implicit user feedback," in *RecSys*, 2013.

[87] X. Han, C. Shi, S. Wang, P. S. Yu, and L. Song, "Aspect-level deep collaborative filtering via heterogeneous information networks," in *IJCAI*, 2018.

[88] Y. Xu, Y. Zhu, Y. Shen, and J. Yu, "Learning shared vertex representation in heterogeneous graphs with convolutional networks for recommendation," in *IJCAI*, 2019.

[89] Z. Wang, H. Liu, Y. Du, Z. Wu, and X. Zhang, "Unified embedding model over heterogeneous information network for personalized recommendation," in *IJCAI*, 2019.

[90] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *KDD*, 2016.

[91] L. Hu, C. Li, C. Shi, C. Yang, and C. Shao, "Graph neural news recommendation with long-term and short-term interest modeling," *Information Processing and Management*, vol. 57, no. 2, p. 102142, 2020.

[92] Z. Liu, M. Wan, S. Guo, K. Achan, and P. S. Yu, "Basconv: Aggregating heterogeneous interactions for basket recommendation with graph convolutional neural network," in *SDM*, 2020.

[93] C. Zhang, C. Huang, L. Yu, X. Zhang, and N. V. Chawla, "Camel: Content-aware and meta-path augmented metric learning for author identification," in *WWW*, 2018.

[94] C. Park, D. Kim, Q. Zhu, J. Han, and H. Yu, "Task-guided pair embedding in heterogeneous network," in *CIKM*, 2019.

[95] Y. Zhang, Y. Fan, Y. Ye, L. Zhao, and C. Shi, "Key player identification in underground forums over attributed heterogeneous information network embedding framework," in *CIKM*, 2019.

[96] Y. Fan, Y. Zhang, S. Hou, L. Chen, Y. Ye, C. Shi, L. Zhao, and S. Xu, "idev: Enhancing social coding security by cross-platform user identification between github and stack overflow," in *IJCAI*, 2019.

[97] Y. Zhang, Y. Fan, W. Song, S. Hou, Y. Ye, X. Li, L. Zhao, C. Shi, J. Wang, and Q. Xiong, "Your style your identity: Leveraging writing and photography styles for drug trafficker identification in darknet markets over attributed heterogeneous information network," in *WWW*, 2019.

[98] G. Jeh and J. Widom, "Scaling personalized web search," in *WWW*, 2003.

[99] C. Shi, X. Kong, Y. Huang, P. S. Yu, and B. Wu, "Hetesim: A general framework for relevance measure in heterogeneous networks," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 10, pp. 2479–2492, 2014.

[100] Z. Liu, V. W. Zheng, Z. Zhao, F. Zhu, K. C.-C. Chang, M. Wu, and J. Ying, "Semantic proximity search on heterogeneous graph by proximity embedding," in *AAAI*, 2017.

[101] Z. Liu, V. W. Zheng, Z. Zhao, H. Yang, K. C. Chang, M. Wu, and J. Ying, "Subgraph-augmented path embedding for semantic user search on heterogeneous social network," in *WWW*, 2018.

[102] Z. Liu, V. W. Zheng, Z. Zhao, F. Zhu, K. C.-C. Chang, M. Wu, and J. Ying, "Distance-aware dag embedding for proximity search on heterogeneous graphs," in *AAAI*, 2018.

[103] R. Bian, Y. S. Koh, G. Dobbie, and A. Divoli, "Network embedding and change modeling in dynamic heterogeneous networks," in *SIGIR*, 2019.

[104] A. M. Fard, E. Bagheri, and K. Wang, "Relationship prediction in dynamic heterogeneous information networks," in *ECIR*, 2019.

[105] H. Xue, L. Yang, W. Jiang, Y. Wei, Y. Hu, and Y. Lin, "Modeling dynamic heterogeneous network for link prediction using hierarchical attention with temporal rnn," *arXiv preprint arXiv:2004.01024*, 2020.

[106] L. Song, Y. Zhang, Z. Wang, and D. Gildea, "A graph-to-sequence model for amr-to-text generation," in *ACL*, 2018.

[107] D. Beck, G. Haffari, and T. Cohn, "Graph-to-sequence learning using gated graph neural networks," in *ACL*, 2018.

[108] S. Yao, T. Wang, and X. Wan, "Heterogeneous graph transformer for graph-to-sequence learning," in *ACL*, 2020.

[109] Y. Ji, C. Shi, F. Zhuang, and P. S. Yu, "Integrating topic model and heterogeneous information network for aspect mining with rating bias," in *PAKDD*, 2019.

[110] J. Zhang, B. Dong, and P. S. Yu, "Deep diffusive neural network based fake news detection from heterogeneous social networks," in *BigData*, 2019.

[111] J. Hu, S. Qian, Q. Fang, and C. Xu, "Hierarchical graph semantic pooling network for multi-modal community question answer matching," in *ACM Multimedia*, 2019.

[112] ——, "Attentive interactive convolutional matching for community question answering in social multimedia," in *ACM Multimedia*, 2018.

[113] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond euclidean data," *IEEE Signal Process. Mag.*, vol. 34, no. 4, pp. 18–42, 2017.

[114] I. Chami, Z. Ying, C. Ré, and J. Leskovec, "Hyperbolic graph convolutional neural networks," in *NeurIPS*, 2019.

[115] Q. Liu, M. Nickel, and D. Kiela, "Hyperbolic graph neural networks," in *NeurIPS*, 2019.

[116] C. Shi, J. Ding, X. Cao, L. Hu, B. Wu, and X. Li, "Entity set expansion in knowledge graph: a heterogeneous information network perspective," *Frontiers Comput. Sci.*, vol. 15, no. 1, p. 151307, 2021.

[117] Z. Liu, M. Wan, S. Guo, K. Achan, and P. S. Yu, "Basconv: Aggregating heterogeneous interactions for basket recommendation with graph convolutional neural network," in *SDM*, 2020.

[118] H. Peng, J. Li, Q. Gong, Y. Song, Y. Ning, K. Lai, and P. S. Yu, "Fine-grained event categorization with heterogeneous graph convolutional networks," in *IJCAI*, 2019.

[119] J. Zhang, C. Xia, C. Zhang, L. Cui, Y. Fu, and P. S. Yu, "BL-MNE: emerging heterogeneous social network embedding through broad learning with aligned autoencoder," in *ICDM*, 2017.

[120] K. Zhao, T. Bai, B. Wu, B. Wang, Y. Zhang, Y. Yang, and J. Nie, "Deep adversarial completion for sparse heterogeneous information network embedding," in *WWW*, 2020.

[121] W. Chen, Y. Gu, Z. Ren, X. He, H. Xie, T. Guo, D. Yin, and Y. Zhang, "Semi-supervised user profiling with heterogeneous graph attention networks." in *IJCAI*, 2019.

[122] 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.

[123] J. Xu, Z. Zhu, J. Zhao, X. Liu, M. Shan, and J. Guo, "Gemini: A novel and universal heterogeneous graph information fusing framework for online recommendations," in *KDD*, 2020.

[124] 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.

[125] A. Li, Z. Qin, R. Liu, Y. Yang, and D. Li, "Spam review detection with graph convolutional networks," in *CIKM*, 2019.

[126] Y. Ye, T. Li, D. Adjeroh, and S. S. Iyengar, "A survey on malware detection using data mining techniques," *ACM Computing Surveys (CSUR)*, vol. 50, no. 3, pp. 1–40, 2017.

[127] A. P. Felt, M. Finifter, E. Chin, S. Hanna, and D. Wagner, "A survey of mobile malware in the wild," in *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, 2011.

[128] S. Wang, Z. Chen, X. Yu, D. Li, J. Ni, L. Tang, J. Gui, Z. Li, H. Chen, and P. S. Yu, "Heterogeneous graph matching networks for unknown malware detection," in *IJCAI*, 2019.

[129] Y. Gao, X. Li, H. Peng, B. Fang, and P. S. Yu, "Hincti: A cyber threat intelligence modeling and identification system based on heterogeneous information network," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[130] Y. Xiao, J. Zhang, and L. Deng, "Prediction of lncrna-protein interactions using hetesim scores based on heterogeneous networks," *Scientific reports*, vol. 7, no. 1, pp. 1–12, 2017.

[131] W. Luo, H. Zhang, X. Yang, L. Bo, X. Yang, Z. Li, X. Qie, and J. Ye, "Dynamic heterogeneous graph neural network for real-time event prediction," in *KDD*, 2020.

[132] H. Hong, Y. Lin, X. Yang, Z. Li, K. Fu, Z. Wang, X. Qie, and J. Ye, "Heteta: Heterogeneous information network embedding for estimating time of arrival," in *KDD*, 2020.

[133] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, "Arnetminer: extraction and mining of academic social networks," in *KDD*, 2008.

[134] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *CoRR*, vol. abs/1611.07308, 2016.

[135] D. Zhu, P. Cui, D. Wang, and W. Zhu, "Deep variational network embedding in wasserstein space," in *KDD*, 2018.

[136] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.

[137] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *ICLR*, 2018.

[138] Q. Li, Z. Han, and X. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *AAAI*, 2018.

[139] X. Liu, F. Zhang, Z. Hou, Z. Wang, L. Mian, J. Zhang, and J. Tang, "Self-supervised learning: Generative or contrastive," *CoRR*, vol. abs/2006.08218, 2020.

[140] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," in *ICLR*, 2019.

[141] K. Sun, Z. Lin, and Z. Zhu, "Multi-stage self-supervised learning for graph convolutional networks on graphs with few labeled nodes," in *AAAI*, 2020.

[142] Z. Peng, Y. Dong, M. Luo, X. Wu, and Q. Zheng, "Self-supervised graph representation learning via global context prediction," *CoRR*, vol. abs/2003.01604, 2020.

[143] Y. You, T. Chen, Z. Wang, and Y. Shen, "When does self-supervision help graph convolutional networks?" *CoRR*, vol. abs/2006.09136, 2020.

[144] Z. Hu, Y. Dong, K. Wang, K. Chang, and Y. Sun, "GPT-GNN: generative pre-training of graph neural networks," in *KDD*, 2020.

[145] J. Qiu, Q. Chen, Y. Dong, J. Zhang, H. Yang, M. Ding, K. Wang, and J. Tang, "GCC: graph contrastive coding for graph neural network pre-training," in *KDD*, 2020.

[146] A. J. Bose and W. L. Hamilton, "Compositional fairness constraints for graph embeddings," in *ICML*, 2019.

[147] M. Du, F. Yang, N. Zou, and X. Hu, "Fairness in deep learning: A computational perspective," *CoRR*, vol. abs/1908.08843, 2019.

[148] T. A. Rahman, B. Surma, M. Backes, and Y. Zhang, "Fairwalk: Towards fair graph embedding," in *IJCAI*, 2019.

[149] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *ICLR*, 2018.

[150] S. Narayanaswamy, B. Paige, J. van de Meent, A. Desmaison, N. D. Goodman, P. Kohli, F. D. Wood, and P. H. S. Torr, "Learning disentangled representations with semi-supervised deep generative models," in *NeurIPS*, 2017.

[151] J. Ma, C. Zhou, P. Cui, H. Yang, and W. Zhu, "Learning disentangled representations for recommendation," in *NeurIPS*, 2019.

[152] K. Tsuyuzaki and I. Nikaido, "Biological systems as heterogeneous information networks: A mini-review and perspectives," *CoRR*, vol. abs/1712.08865, 2017.

[153] S. Ji, S. Pan, E. Cambria, P. Marttinen, and P. S. Yu, "A survey on knowledge graphs: Representation, acquisition and applications," *CoRR*, vol. abs/2002.00388, 2020.

[154] Q. Guo, F. Zhuang, C. Qin, H. Zhu, X. Xie, H. Xiong, and Q. He, "A survey on knowledge graph-based recommender systems," *CoRR*, vol. abs/2003.00911, 2020.

[155] H. Wang, M. Zhao, X. Xie, W. Li, and M. Guo, "Knowledge graph convolutional networks for recommender systems," in *WWW*, 2019.