Chapter 16
Heterogeneous Graph Neural Networks

Chuan Shi

Abstract Heterogeneous graphs (HGs) also called heterogeneous information networks (HINs) have become ubiquitous in real-world scenarios. Recently, employing graph neural networks (GNNs) to heterogeneous graphs, known as heterogeneous graph neural networks (HGNNs) which aim to learn embedding in low-dimensional space while preserving heterogeneous structure and semantic for downstream tasks, has drawn considerable attention. This chapter will first give a brief review of the recent development on HG embedding, then introduce typical methods from the perspective of shallow and deep models, especially HGNNs. Finally, it will point out future research directions for HGNNs.

16.1 Introduction to HGNNs

Heterogeneous graphs (HGs) (Sun and Han, 2013), which compose different types of entities and relations, also known as heterogeneous information networks (HINs), are ubiquitous in real-world scenarios, ranging from bibliographic networks, social networks to recommender systems. For example, as shown in Fig. 16.1 (a), a bibliographic network can be represented by a HG, which consists of four types of entities (author, paper, venue, and term) and three types of relations (author-write-paper, paper-contain-term and conference-publish-paper); and these basic relations can be further derived for more complex semantics (e.g., author-write-paper-contain-item). It has been well recognized that HG is a powerful model that embraces rich semantic and structural information. Therefore, researches on HG have been experiencing tremendous growth in data mining and machine learning, many of which have successful applications such as recommendation (Shi et al, 2018a; Hu et al, 2018a), text

Chuan Shi
School of Computer Science, Beijing University of Posts and Telecommunications, e-mail: shichuan@bupt.edu.cn

analysis (Linmei et al, 2019; Hu et al, 2020a), and cybersecurity (Hu et al, 2019b; Hou et al, 2017).

Due to the ubiquity of HGs, how to learn embedding of HGs is a key research problem in various graph analysis applications, e.g., node/graph classification (Dong et al, 2017; Fu et al, 2017), and node clustering (Li et al, 2019g). Traditionally, matrix factorization methods (Newman, 2006b) generate latent features in HGs. However, the computational cost of decomposing a large-scale matrix is usually very expensive, and also suffers from its statistical performance drawback (Shi et al, 2016; Cui et al, 2018). To address this challenge, heterogeneous graph embedding, aiming to learn a function that maps input space into lower-dimensional space while preserving heterogeneous structure and semantic, has drawn considerable attention in recent years.

Although there have been ample studies of embedding technology on homogeneous graphs (Cui et al, 2018) which consist of only one type of nodes and edges, these techniques cannot be directly applicable to HGs due to heterogeneity. Specifically, (1) the structure in HGs is usually semantic dependent, e.g., meta-path structure (Dong et al, 2017) can be very different when considering different types of relations; (2) different types of nodes and edges have different attributes located in different feature spaces; (3) HGs are usually application dependent, which may need sufficient domain knowledge for meta-path/meta-graph selection.

To tackle the above issues, various HG embedding methods have been proposed (Chen et al, 2018b; Hu et al, 2019a; Dong et al, 2017; Fu et al, 2017; Wang et al, 2019m; Shi et al, 2018a; Wang et al, 2020n). From the technical perspective, we divide the widely used models in HG embedding into two categories: shallow models and deep models. In summary, shallow models initialize the node embeddings randomly, then learn the node embeddings through optimizing some well-designed objective functions to preserve heterogeneous structures and semantics. Deep model aims to use deep neural networks (DNNs) to learn embedding from node attributes or interactions, where heterogeneous graph neural networks (HGNNs) stand out and will be the focus of this chapter. And there have demonstrated the success of HG embedding techniques deployed in real-world applications including recommender systems (Shi et al, 2018a; Hu et al, 2018a; Wang et al, 2020n), malware detection systems (Hou et al, 2017; Fan et al, 2018; Ye et al, 2019a), and healthcare systems (Cao et al, 2020; Hosseini et al, 2018).

The remainder of this chapter is organized as follows. In Sect. 27.1, we first introduce basic concepts in HGs, then discuss unique challenges of HG embedding due to the heterogeneity and give a brief review of the recent development on HG embedding. In Sect. 24.2 and 20.3, we categorize and introduce HG embedding in details according to the shallow and deep models. In Sect. 20.4, we further review pros and cons of the models introduced above. Finally, Sect. 20.5 forecasts the future research directions for HGNNs.

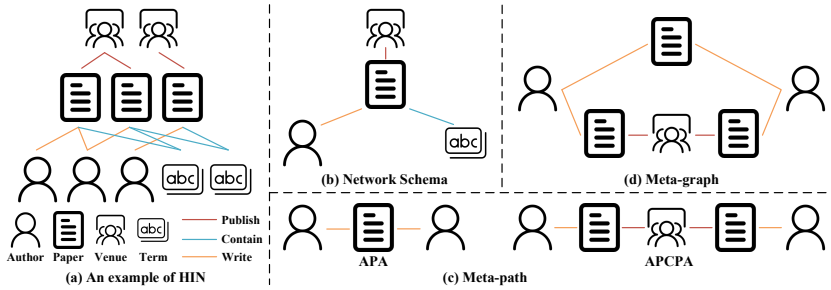

Fig. 16.1: An illustrative example of a heterogeneous graph (Wang et al, 2020l). (a) A bibliographic graph including four types of entities (i.e., author, paper, venue and term) and three types of relations (i.e., publish, contain and write). (b) Network schema of the bibliographic graph. (c) Two meta-paths (i.e., author-paper-author and paper-term-paper). (d) A meta-graph used in the bibliographic graph.

16.1.1 Basic Concepts of Heterogeneous Graphs

In this section, we will first formally introduce basic concepts in HGs and illustrate the symbols used throughout this chapter. 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 16.1. Heterogeneous Graph (or Heterogeneous Information Network) (Sun and Han, 2013). A HG is defined as a graph $\mathscr{G} = \{\mathscr{V}, \mathscr{E}\}$, in which $\mathscr{V}$ and $\mathscr{E}$ represent the node set and the edge set, respectively. Each node $v \in \mathscr{V}$ and each edge $e \in \mathscr{E}$ are associated with their mapping function $\phi(v) : \mathscr{V} \to \mathscr{A}$ and $\varphi(e) : \mathscr{E} \to \mathscr{R}$. $\mathscr{A}$ and $\mathscr{R}$ denote the node type set and edge type set, respectively, where $|\mathscr{A}| + |\mathscr{R}| > 2$. The **network schema** for $\mathscr{G}$ is defined as $\mathscr{S} = (\mathscr{A}, \mathscr{R})$, which can be seen as a meta template of a heterogeneous graph $\mathscr{G} = \{\mathscr{V}, \mathscr{E}\}$ with the node type mapping function $\phi(v) : \mathscr{V} \to \mathscr{A}$ and the edge type mapping function $\varphi(e) : \mathscr{E} \to \mathscr{R}$. The network schema is a graph defined over node types $\mathscr{A}$, with edges as relation types from $\mathscr{R}$.

HG not only provides graph structure of data association, but also portrays higher-level semantics. An example of HG is illustrated in Fig. 16.1 (a), which consists of four node types (author, paper, venue, and term) and three edge types (author-write-paper, paper-contain-term, and conference-publish-paper), and Fig. 16.1 (b) illustrates the network schema. To formulate semantics of higher-order relationships among entities, meta-path (Sun et al, 2011) is further proposed whose definition is given below.

Definition 16.2. Meta-path (Sun et al, 2011). A meta-path p is based on network schema $\mathscr{S}$, which is denoted as $p = N_1 \xrightarrow{R_1} N_2 \xrightarrow{R_2} \cdots \xrightarrow{R_l} N_{l+1}$ (simplified to

$N_1N_2\cdots N_{l+1}$) with node types $N_1, N_2, \cdots, N_{l+1} \in \mathscr{N}$ and edge types $R_1, R_2, \cdots R_l \in \mathscr{R}$.

Note that different meta-paths describe semantic relationships in different views. For example, the meta-path *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 (Milo et al, 2002). To address this challenge, meta-graph (Huang et al, 2016b) is proposed to use a directed acyclic graph of entity and relation types to capture more complex relationships between entities, defined as follows.

Definition 16.3. Meta-graph (Huang et al, 2016b). A meta-graph $\mathscr{T}$ can be seen as a directed acyclic graph (DAG) composed of multiple meta-paths with common nodes. Formally, meta-graph is defined as $\mathscr{T} = (\mathscr{V}_{\mathscr{T}}, \mathscr{E}_{\mathscr{T}})$, where $\mathscr{V}_{\mathscr{T}}$ is a set of nodes and $\mathscr{E}_{\mathscr{T}}$ is a set of edges. For any node $v \in \mathscr{V}_{\mathscr{T}}, \phi(v) \in \mathscr{A}$; for any edge $e \in \mathscr{E}_{\mathscr{T}}, \varphi(e) \in \mathscr{R}$.

An example meta-graph is shown in Fig. 16.1 (d), which can be regarded as the combination of meta-path *APA* and *APCPA*, reflecting high-order similarity of two nodes. Note that a meta-graph can be symmetric or asymmetric (Zhang et al, 2020g). To learn embeddings of HG, we formalize the problem of heterogeneous graph embedding.

Definition 16.4. Heterogeneous Graph Embedding (Shi et al, 2016). Heterogeneous graph embedding aims to learn a function $\Phi : \mathscr{V} \to \mathbb{R}^d$ that embeds the nodes $v \in \mathscr{V}$ in HG into low-dimensional Euclidean space with $d \ll |\mathscr{V}|$.

16.1.2 Challenges of HG Embedding

Different from homogeneous graph embedding (Cui et al, 2018), where the basic problem is preserving structure and property in node embedding (Cui et al, 2018). Due to the heterogeneity, HG 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 first-order, second-order, and even higher-order structures (Tang et al, 2015b). All these structures are well defined and have good intuition. However, the structure in HGs will dramatically change depending on the selected relations. Let's still take the academic graph in Fig. 16.1 (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 higher-order structures in HGs, will result in different and more complicated structures. Therefore, how to efficiently and effectively preserve these complex structures is of great challenge in HG embedding,

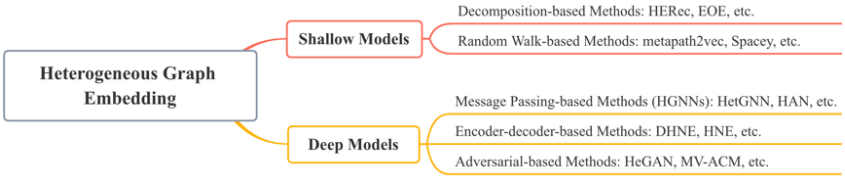

Fig. 16.2: Heterogeneous graph embedding tree classification diagram.

while current efforts have been made towards the meta-path structure (Dong et al, 2017) and meta-graph structure (Zhang et al, 2018b).

Heterogeneous Attributes (the fusion problem caused by the heterogeneity of attributes). Since 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 attributes of its neighbors. However, in HGs, the attributes of different types of nodes and edges may have different meanings (Zhang et al, 2019b; Wang et al, 2019m). For example, the attributes of author can be 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 HG 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 HGs. 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 side information (e.g., node attributes) (Wang et al, 2019m; Zhang et al, 2019b) or more advanced domain knowledge (Shi et al, 2018a; Chen and Sun, 2017) to HG embedding process.

16.1.3 Brief Overview of Current Development

Most of early works on graph data are based on high-dimensional sparse vectors for matrix analysis. However, the sparsity of the graph in reality and its growing scale have created serious challenges for such methods. A more effective way is to map nodes to latent space and use low-dimensional vectors to represent them. Therefore, they can be more flexibly applied to different data mining tasks, i.e., graph embedding.

There has been a lot of works dedicated to homogeneous graph embedding (Cui et al, 2018). These works are mainly based on deep models and combined with graph properties to learn embeddings of nodes or edges. For instance, DeepWalk (Perozzi

et al, 2014) combines random walk and skip-gram model; LINE (Tang et al, 2015b) utilizes first-order and second-order similarity to learn distinguished node embedding for large-scale graphs; SDNE (Wang et al, 2016) uses deep auto-encoders to extract non-linear characteristics of graph structure. In addition to structural information, many methods further use the content of nodes or other auxiliary information (such as text, images, and tags) to learn more accurate and meaningful node embeddings. Some survey papers comprehensively summarize the work in this area (Cui et al, 2018; Hamilton et al, 2017c).

Due to the heterogeneity, embedding techniques for homogeneous graphs cannot be directly applicable to HGs. Therefore, researchers have begun to explore HG embedding methods, which emerge in recent years but develop rapidly. From the technical perspective, we summarize the widely used techniques (or models) in HG embedding, which can be generally divided into two categories: shallow models and deep models, as shown in Fig. 16.2. Specifically, shallows model mainly rely on meta-paths to simplify the complex structure of HGs, which can be classified into decomposition-based and random walk-based. Decomposition-based techniques Chen et al (2018b); Xu et al (2017b); Shi et al (2018b,c); Matsuno and Murata (2018); Tang et al (2015a); Gui et al (2016) decompose complex heterogeneous structure into several simpler homogeneous structures; while random walk-based (Dong et al, 2017; Hussein et al, 2018) methods utilize meta-path-guided random walk to preserve specific first-order and high-order structures. In order to take full advantage of heterogeneous structures and attributes, deep models are threefold: message passing-based (HGNNs), encoder-decoder-based and adversarial-based methods. Message passing mechanism, i.e., the core idea of graph neural networks (GNNs), seamlessly integrates structure and attribute information. HGNNs inherit the message passing mechanism and design suitable aggregation functions to capture rich semantic in HGs (Wang et al, 2019m; Fu et al, 2020; Hong et al, 2020b; Zhang et al, 2019b; Cen et al, 2019; Zhao et al, 2020b; Zhu et al, 2019d; Schlichtkrull et al, 2018). The remaining encoder-decoder-based (Tu et al, 2018; Chang et al, 2015; Zhang et al, 2019c; Chen and Sun, 2017) and adversarial-based (Hu et al, 2018a; Zhao et al, 2020c) techniques employ encoder-decoder framework or adversarial learning to preserve complex attribute and structural information of HGs. In the following sections, we will introduce representative works of their subcategories in detail and compare their pros and cons.

16.2 Shallow Models

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

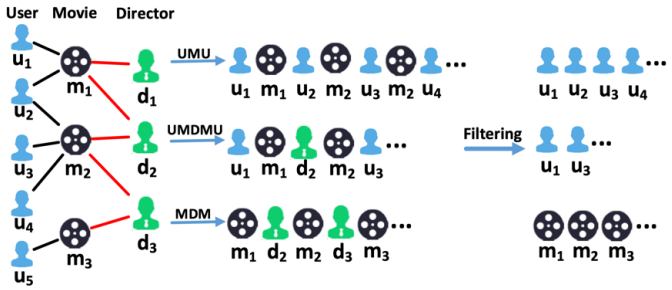

Fig. 16.3: An illustrative example of the proposed meta-path-guided random walk in HERec (Shi et al, 2018a). HERec first perform random walks guided by some selected meta-paths, then filter node sequences not with the user type or item type.

16.2.1 Decomposition-based Methods

To cope with the challenges brought by heterogeneity, decomposition-based techniques (Chen et al, 2018b; Xu et al, 2017b; Shi et al, 2018b,c; Matsuno and Murata, 2018; Tang et al, 2015a; Gui et al, 2016) decompose HG into several simpler subgraphs and preserve the proximity of nodes in each sub-graph, finally merge the information to achieve the effect of divide and conquer.

Specifically, HERec (Shi et al, 2018a) aims to learn 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, as shown in Fig. 16.3. Then it uses node2vec (Grover and Leskovec, 2016) to learn preliminary embeddings from the co-occurrence sequences of users and items. Because embeddings under different meta-paths contain different semantic information, for better recommendation performance, HERec designs a fusion function to unify the multiple embeddings:

$$g(\mathbf{h}_u^p) = \frac{1}{|P|} \sum_{p=1}^{P} (W^p \mathbf{h}_u^p + \mathbf{b}^p), \tag{16.1}$$

where $\mathbf{h}_u^p$ is the embedding of user node u in meta-path p. P 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.

As another example, EOE is proposed to learn embeddings for coupled HGs, which consist of two different but related subgraphs. It divides the edges in HG into intra-graph edges and inter-graph edges. Intra-graph edge connects two nodes with the same type, and inter-graph edge connects two nodes with different types. To capture the heterogeneity in inter-graph edge, EOE (Xu et al, 2017b) uses the relation-specific matrix M_r to calculate the similarity between two nodes, which can be formulated as:

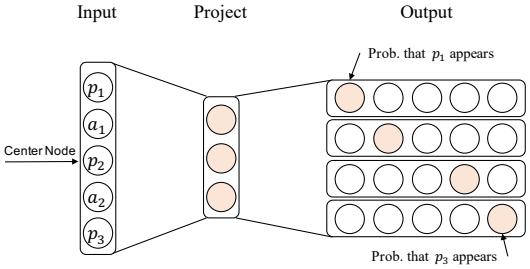

Fig. 16.4 The architecture of metapath2vec (Dong et al, 2017). Node sequence is generated under the meta-path PAP. It projects the embedding of the center node, e.g., p_2 into latent space and maximizes the probability of its meta-path-based context nodes, e.g., p_1, p_3, a_1 and a_2, appearing.

$$S_r(v_i, v_j) = \frac{1}{1+\exp\left\{-\mathbf{h}_i^\top M_r \mathbf{h}_j\right\}}. \tag{16.2}$$

Similarly, PME (Chen et al, 2018b) decomposes HG into some bipartite graphs according to the types of edges and projects each bipartite graph into a relation-specific semantic space. PTE (Tang et al, 2015a) divides the documents into word-word graph, word-document graph and word-label graph. Then it uses LINE (Tang et al, 2015b) to learn the shared node embeddings for each sub-graph. HEBE (Gui et al, 2016) samples a series of subgraphs from a HG and preserves the proximity between the center node and its subgraph.

The above-mentioned two-step framework of decomposition and fusion, as a transition product from homogeneous networks to HGs, is often used in the early attempt of HG embedding. Later, researchers gradually realized that extracting homogeneous graphs from HGs would irreversibly lose information carried by heterogeneous neighbors, and began to explore HG embedding methods that truly adapted to heterogeneous structure.

16.2.2 Random Walk-based Methods

Random walk, which generates some node sequences in a graph, is often used to describe the reachability between nodes. Therefore, it is widely used in graph representation learning to sample neighbor relationships of nodes and capture local structure in the graph (Grover and Leskovec, 2016). In homogeneous graphs, the node type is single and random walk can walk along any path. While in HGs, due to the type constraints of nodes and edges, meta-path-guided random walk is usually adopted, so that the generated node sequence contains not only the structural information, but also the semantic information. Through preserving the node sequence structure, node embedding can preserve both first-order and high-order proximity (Dong et al, 2017). A representative work is metapath2vec (Dong et al, 2017), which uses meta-path-guided random walk to capture semantic information of two nodes, e.g., the co-author relationship in academic graph as shown in Fig. 16.4.

Metapath2vec (Dong et al, 2017) mainly uses meta-path-guided random walk to generate heterogeneous node sequences with rich semantic, then it designs a het-

erogeneous skip-gram technique to preserve the proximity between node v and its context nodes, i.e., neighbors in the random walk sequences:

$$\arg\max_{\theta} \sum_{v\in\mathcal{V}} \sum_{t\in\mathcal{N}} \sum_{c_t\in C_t(v)} \log p(c_t|v;\theta), \tag{16.3}$$

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:

$$p(c_t|v;\theta) = \frac{e^{\mathbf{h}_{c_t}\cdot\mathbf{h}_v}}{\sum_{\tilde{v}\in\mathcal{V}} e^{\mathbf{h}_{\tilde{v}}\cdot\mathbf{h}_v}}. \tag{16.4}$$

From the diagram shown in Fig. 16.4, Eq. (16.4) needs to calculate similarity between center node and its neighbors. Then Mikolov et al (2013b) introduces a negative sampling strategy to reduce the computation. Hence, Eq. (16.4) can be approximated as:

$$\log\sigma(\mathbf{h}_{c_t}\cdot\mathbf{h}_v) + \sum_{q=1}^{Q} \mathbb{E}_{\tilde{v}^q\sim P(\tilde{v})}\left[\log\sigma\left(-\mathbf{h}_{\tilde{v}^q}\cdot\mathbf{h}_v\right)\right], \tag{16.5}$$

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. Through the strategy of negative sampling, the time complexity is greatly reduced. 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 $P(\tilde{v})$. Thus it further designs metapath2vec++, which samples negative nodes of the same type as the central node, i.e., $\tilde{v}_t^q \sim P(\tilde{v}_t)$. The formulation can be rewritten as:

$$\log\sigma(\mathbf{h}_{c_t}\cdot\mathbf{h}_v) + \sum_{q=1}^{Q} \mathbb{E}_{\tilde{v}_t^q\sim P(\tilde{v}_t)}\left[\log\sigma\left(-\mathbf{h}_{\tilde{v}_t^q}\cdot\mathbf{h}_v\right)\right]. \tag{16.6}$$

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 (He et al, 2019) designs a heterogeneous spacey random walk to unify different metapaths with a second-order hyper-matrix to control transition probability among different node types. JUST (Hussein et al, 2018) 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 (Lee et al, 2019e) proposes an extended skip-gram technique to balance the various types of relations. It treats heterogeneous graph embedding as multiple relation-based tasks, and balances the influence of different relations on node embeddings by adjusting the training ratio of different tasks. HHNE (Wang et al, 2019n) conducts meta-path-guided random walk in hyperbolic space (Helgason, 1979), where the similarity between nodes can be measured using hyperbolic distance. In this way, some properties of

HGs, e.g., hierarchical and power-law structure, can be naturally reflected in learned node embeddings.

16.3 Deep Models

In recent years, deep neural networks (DNNs) have achieved great success in the fields of computer vision and natural language processing. Some works have also begun to use deep models to learn embedding from node attributes or interactions among nodes in HGs. Compared with shallow models, deep models can better capture the non-linear relationship, which can be roughly divided into three categories: message passing-based, encoder-decoder-based and adversarial-based.

16.3.1 Message Passing-based Methods (HGNNs)

Graph neural networks (GNNs) have emerged recently. Its core idea is the message passing mechanism, which aggregates neighborhood information and transmits it as messages to neighbor nodes. Different from GNNs that can directly fuse attributes of neighbors to update node embeddings, due to different types of nodes and edges, HGNNs need to overcome the heterogeneity of attributes and design effective fusion methods to utilize neighborhood information. Therefore, the key component is to design a suitable aggregation function, which can capture semantic and structural information of HGs (Wang et al, 2019m; Fu et al, 2020; Hong et al, 2020b; Zhang et al, 2019b; Cen et al, 2019; Zhao et al, 2020b; Zhu et al, 2019d; Schlichtkrull et al, 2018).

Unsupervised HGNNs. Unsupervised HGNNs aim to learn node embeddings with good generalization. To this end, they always utilize interactions among different types of attributes to capture the potential commonalities. HetGNN (Zhang et al, 2019b) 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:

$$f_1(v) = \frac{\sum_{i \in C_v} [\overrightarrow{LSTM}\{\mathscr{FC}(\mathbf{h}_i)\} \oplus \overleftarrow{LSTM}\{\mathscr{FC}(\mathbf{h}_i)\}]}{|C_v|}, \tag{16.7}$$

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) (Huang et al, 2015) is used to fuse the embeddings learned by multiple attribute encoder $\mathscr{FC}$. Neighbor aggregation aims to aggregate the nodes with same type by using a Bi-LSTM to capture the position information:

$$f_2^t(v) = \frac{\sum_{v' \in N_t(v)} [\overrightarrow{LSTM}\{f_1(v')\} \oplus \overleftarrow{LSTM}\{f_1(v')\}]}{|N_t(v)|}, \tag{16.8}$$

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.

$$\mathbf{h}_v = \alpha^{v,v} f_1(v) + \sum_{t \in O_v} \alpha^{v,t} f_2^t(v). \tag{16.9}$$

where $\mathbf{h}_v$ is the final embedding of node v, and O_v denotes the set of node types. Finally, a heterogeneous skip-gram loss is used as the unsupervised graph context loss to update node embeddings. Through these three aggregation methods, HetGNN can preserve the heterogeneity of both graph structures and node attributes.

Other unsupervised methods capture either heterogeneity of node attributes or heterogeneity of graph structures. HNE (Chang et al, 2015) is proposed to learn embeddings for the cross-model data in HGs, but it ignores the various types of edges. SHNE (Zhang et al, 2019c) focuses on capturing semantic information of nodes by designing a deep semantic encoder with gated recurrent units (GRU) (Chung et al, 2014). Although it uses heterogeneous skip-gram to preserve the heterogeneity of graph, SHNE is designed specifically for text data. Cen proposes GATNE (Cen et al, 2019), 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 edge relationships between node pairs.

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 the attention mechanism to capture the most relevant structural and attribute information to the task. Wang (Wang et al, 2019m) 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. 16.5.

It consists of three parts: node-level attention, semantic-level attention, and prediction. Node-level attention aims to utilize the self-attention mechanism (Vaswani et al, 2017) to learn importances of neighbors in a certain meta-path:

$$\alpha_{ij}^m = \frac{\exp(\sigma(\mathbf{a}_m^{\mathrm{T}} \cdot [\mathbf{h}_i' \| \mathbf{h}_j']))}{\sum_{k \in \mathcal{N}_i^m} \exp(\sigma(\mathbf{a}_m^{\mathrm{T}} \cdot [\mathbf{h}_i' \| \mathbf{h}_k']))}, \tag{16.10}$$

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

$$\mathbf{h}_i^m = \sigma\left(\sum_{j \in \mathcal{N}_i^m} \alpha_{ij}^m \cdot \mathbf{h}_j\right), \tag{16.11}$$

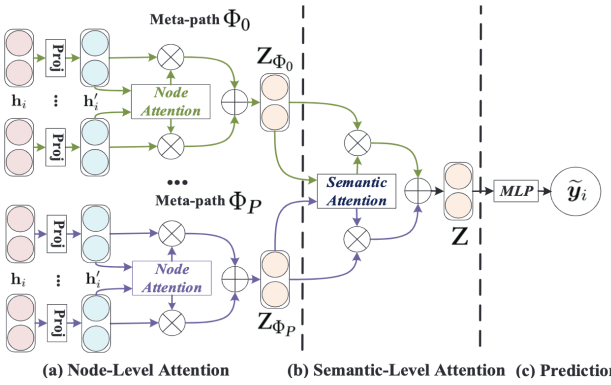

Fig. 16.5: The architecture of HAN (Wang et al, 2019m). The whole model can be divided into three parts: Node-Level Attention aims to learn the importance of neighbors' features. Semantic-Level Attention aims to learn the importance of different meta-paths. Prediction layer utilizes the labeled nodes to update node embeddings.

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 $\{H_{m_0}, H_{m_1}, \cdots, H_{m_P}\}$. To effectively aggregate different semantic embeddings, HAN designs a semantic-level attention mechanism:

$$w_{m_i} = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \mathbf{q}^{\mathrm{T}} \cdot \tanh(W \cdot \mathbf{h}_i^m + \mathbf{b}), \tag{16.12}$$

where $W \in \mathbb{R}^{d' \times d}$ 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:

$$H = \sum_{i=1}^{P} \beta_{m_i} \cdot H_{m_i}, \tag{16.13}$$

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

Subsequently, a series of attention-based HGNNs was proposed (Fu et al, 2020; Hong et al, 2020b; Hu et al, 2020e). MAGNN (Fu et al, 2020) 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 (Hong et al, 2020b) and HGT (Hu et al, 2020e) 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 interactions among different types of nodes, but also assign different weights to neighbors during aggregation.

In addition, there are some HGNNs that focus on other issues. NSHE (Zhao et al, 2020b) proposes to incorporate network schema, instead of meta-path, in aggregating neighborhood information. GTN (Yun et al, 2019) aims to automatically identify the useful meta-paths and high-order edges in the process of learning node embeddings. RSHN (Zhu et al, 2019d) uses both original node graph and coarsened line graph to design a relation-structure aware HGNN. RGCN (Schlichtkrull et al, 2018) uses multiple weight matrices to project the node embeddings into different relation spaces, thus capturing the heterogeneity of the graph.

Compared with shallow models, HGNNs have an obvious advantage that they have the ability of inductive learning, i.e., learning embeddings for out-of-sample nodes. 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.

16.3.2 Encoder-decoder-based Methods

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 (Tu et al, 2018; Chang et al, 2015; Zhang et al, 2019c; Chen and Sun, 2017; Zhang et al, 2018a; Park et al, 2019).

For example, DHNE (Tu et al, 2018) proposes hyper-path-based random walk to preserve both structural information and indecomposability of hyper-graphs. 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. As shown in Fig. 16.6, 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 (Tang et al, 2015b). The second layer is a fully connected layer with embedding concatenated:

$$L = \sigma(W_a\mathbf{h}_a \oplus W_b\mathbf{h}_b \oplus W_c\mathbf{h}_c), \tag{16.14}$$

where 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. W_a, W_b and $W_c \in \mathbb{R}^{d'\times d}$ are the transformation matrices for different node types. Finally, the third layer is used

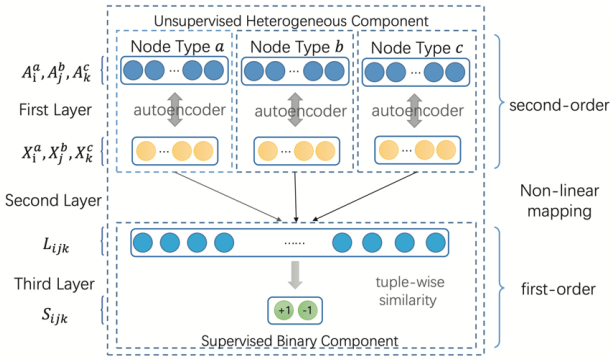

Fig. 16.6: The framework of DHNE (Tu et al, 2018). DHNE learns embeddings for nodes in heterogeneous hypernetworks, which can simultaneously address indecomposable hyperedges while preserving rich structural information.

to calculate the indecomposability of hyperedge:

$$\mathscr{S} = \sigma(W \cdot L + \mathbf{b}), \tag{16.15}$$

where $\mathscr{S}$ denote the indecomposability of hyperedge; $W \in \mathbb{R}^{1\times 3d'}$ and $\mathbf{b} \in \mathbb{R}^{1\times 1}$ are the weight matrix and bias, respectively. A higher value of $\mathscr{S}$ means these nodes are from the existing hyperedges, otherwise it should be small.

Similarly, HNE (Chang et al, 2015) focuses on multi-modal heterogeneous graph. It uses CNN and autoencoder to learn embedding from images and texts, respectively. Then it uses the embedding to predict whether there is an edge between the images and texts. Camel (Zhang et al, 2018a) uses GRU as an encoder to learn paper embedding from the abstracts. A skip-gram objective function is used to preserve the local structures of the graphs.

16.3.3 Adversarial-based Methods

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 information, for example, GraphGAN (Wang et al, 2018a) 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 (Hu et al, 2018a) is the first to use GAN in HG embedding. It incorporates the multiple relations into the generator and discriminator, so that the heterogeneity of a given graph can be considered.

As shown in Fig. 16.7 (c), HeGAN mainly consists of two competing players, the discriminator and the generator. Given a node, the generator attempts to produce

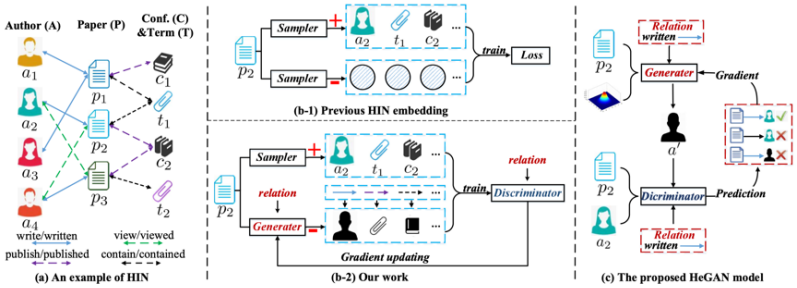

Fig. 16.7: Overview of HeGAN (Hu et al, 2018a). (a) A toy HG for bibliographic data. (b) Comparison between HeGAN and previous works. (c) The framework of HeGAN for adversarial learning on HGs.

fake samples associated with the given node to feed into the discriminator, whereas the discriminator tries to improve its parameterization to separate the fake samples from the real ones actually connected to the given node. The better trained discriminator would then force the generator to produce better fake samples, and the process is repeated. During such iterations, both the generator and discriminator receive mutual, positive reinforcement. While this setup may appear similar to previous efforts (Cai et al, 2018c; Dai et al, 2018c; Pan et al, 2018) on GAN-based network embedding, HeGAN employs two major novelties to address the challenges of adversarial learning on HINs.

First, existing studies only leverage GAN to distinguish whether a node is real or fake *w.r.t.* structural connections to a given node, without accounting for the heterogeneity in HINs. For example, given a paper p_2, they treat nodes a_2, a_4 as real, whereas a_1, a_3 are fake simply based on the topology of the HIN shown in Fig. 16.7 (a). However, a_2 and a_4 are connected to p_2 for different reasons: a_2 writes p_2 and a_4 only views p_2. Thus, they miss out on valuable semantics carried by HGs, unable to differentiate a_2 and a_4 even though they play distinct semantic roles. Given a paper p_2 as well as a relation, say, write/written, HeGAN introduces a relation-aware discriminator to tell apart a_2 and a_4. Formally, relation-aware discriminator $C(\mathbf{e}_v \mid u, r; \theta^C)$ evaluates the connectivity between the pair of nodes u and v *w.r.t.* a relation r:

$$C(\mathbf{e}_v \mid u, r; \theta^C) = \frac{1}{1+\exp(-\mathbf{e}_u^{C\top} M_r^C \mathbf{e}_v)}, \tag{16.16}$$

where $\mathbf{e}_v \in \mathbb{R}^{d\times 1}$ is the input embedding of the sample v, $\mathbf{e}_u \in \mathbb{R}^{d\times 1}$ is the learnable embedding of node u, and $M_r^C \in \mathbb{R}^{d\times d}$ is a learnable relation matrix for relation r.

Second, existing studies are limited in sample generation in both effectiveness and efficiency. They typically model the distribution of nodes using some form of softmax over all nodes in the original graph. In terms of effectiveness, their fake samples are constrained to the nodes in the graph, whereas the most representative fake samples may fall "in between" the existing nodes in the embedding space. For example, given a paper p_2, they can only choose fake samples from $\mathcal{V}$, such as

a_1 and a_3. However, both may not be adequately similar to real samples such as a_2. Towards a better sample generation, we introduce a generalized generator that can produce latent nodes such as a' shown in Fig. 16.7 (c), where it is possible that $a' \notin \mathcal{V}$. In particular, the generalized generator leverage the following Gaussian distribution:

$$\mathcal{N}(\mathbf{e}_u^{G^\top} M_r^G, \sigma^2 I), \tag{16.17}$$

where $\mathbf{e}_u^G \in \mathbb{R}^{d\times 1}$ and $M_r^G \in \mathbb{R}^{d\times d}$ denote the node embedding of $u \in \mathcal{V}$ and the relation matrix of $r \in \mathcal{R}$ for the generator.

Except for HeGAN, MV-ACM (Zhao et al, 2020c) uses GAN to generate the complementary views by computing the similarity of nodes in different views. Overall, adversarial-based methods prefer to utilize the negative samples to enhance the robustness of embeddings. But the choice of negative samples has a huge influence on the performance, thus leading higher variances.

16.4 Review

Based on the above representative work of the shallow and deep models, it can be found that the shallow models mainly focus on the structure of HGs, and rarely use additional information such as attributes. One of the possible reasons is that shallow models are hard to depict the relationship between additional and structural information. The learning ability of DNNs supports modeling of this complex relationship. For example, message passing-based techniques are good at encoding structures and attributes simultaneously, and integrate 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 functions through different decoders. Adversarial-based methods prefer to utilize the negative samples to enhance the robustness of embeddings. But the choice of negative samples has a huge influence on the performance, thus leading higher variances (Hu et al, 2019a).

However, shallow and deep models each have their own pros and cons. Shallow models lack non-linear representation capability, but are efficient and easy to parallelize. Specially, the complexity of random walk technique consists of two parts: random walk and skip-gram, both of which are linear with the number of nodes. Decomposition technique needs to divide HGs into sub-graphs according to the type of edges, so the complexity is linear with the number of edges, which is higher than random walk. Deep models have stronger representation capability, but they are easier to fit noise and have higher time and space complexity. Additionally, the cumbersome hyperparameter adjustment of deep models is also criticized. But with the popularity of deep learning, deep models, especially HGNNs, have become the main research direction in HG embedding.

16.5 Future Directions

HGNNs have 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.

16.5.1 Structures and Properties Preservation

The basic success of HGNNs builds on the HG structure preservation. This also motivates many HGNNs to exploit different HG structures, where the most typical one is meta-path (Dong et al, 2017; Shi et al, 2016). Following this line, meta-graph structure is naturally considered (Zhang et al, 2018b). However, HG is far more than these structures. Selecting the most appropriate meta-path is still very challenging in the real world. An improper meta-path will fundamentally hinder the performance of HGNNs. Whether we can explore other techniques, e.g., motif (Zhao et al, 2019a; Huang et al, 2016b) or network schema (Zhao et al, 2020b) to capture HG structure is worth pursuing. Moreover, if we rethink the goal of traditional graph embedding, i.e., replacing structure information with the distance/similarity in a metric space, a research direction to explore is whether we can design HGNNs which can naturally learn such distance/similarity rather than using pre-defined meta-path/meta-graph.

As mentioned before, many current HGNNs mainly take the structures into account. However, some properties, which usually provide additional useful information to model HGs, have not been fully considered. One typical property is the dynamics of HG, i.e., a real-world HG always evolves over time. Despite that the incremental learning on dynamic HG is proposed (Wang et al, 2020m), dynamic heterogeneous graph embedding is still facing big challenges. For example, Bian et al (2019) is only proposed with a shallow model, which greatly limits its embedding ability. How can we learn dynamic heterogeneous graph embedding in HGNNs 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 (Kipf and Welling, 2016; Zhu et al, 2018), which is largely ignored by current HGNNs. This suggests a huge potential direction for improving HGNNs.

16.5.2 Deeper Exploration

We have witnessed the great success and large impact of GNNs, where most of the existing GNNs are proposed for homogeneous graph (Kipf and Welling, 2017b; Veličković et al, 2018). Recently, HGNNs have attracted considerable attention (Wang et al, 2019m; Zhang et al, 2019b; Fu et al, 2020; Cen et al, 2019).

One natural question may arise that what is the essential difference between GNNs and HGNNs. More theoretical analysis on HGNNs is seriously lacking. For example, it is well accepted that the GNNs suffer from over-smoothing problem (Li et al, 2018b), so will HGNNs also have such a problem? If the answer is yes, what factor causes the over-smoothing problem in HGNNs since they usually contain multiple aggregation strategies (Wang et al, 2019m; Zhang et al, 2019b).

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 neural networks, thus reducing the dependence on manual labels (Liu et al, 2020f). 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 (Veličković et al, 2018; Sun et al, 2020c). Therefore, exploring self-supervised learning on HGNNs is expected to further facilitate the development of this area.

Another important direction is the pre-training of HGNNs (Hu et al, 2020d; Qiu et al, 2020a). Nowadays, HGNNs are designed independently, i.e., the proposed method usually works well for certain tasks, but the transfer ability across different tasks is ill-considered. When dealing with a new HG or task, we have to train HGNNs from scratch, which is time-consuming and requires a large amount of labels. In this situation, if there is a well pre-trained HGNN with strong generalization that can be fine-tuned with few labels, the time and label consumption can be reduced.

16.5.3 Reliability

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

Fairness. The embeddings learned by methods are sometimes highly related to certain attributes, e.g., age or gender, which may amplify societal stereotypes in the prediction results (Du et al, 2020). Therefore, learning fair or de-biased embeddings is an important research direction. There are some researches on the fairness of homogeneous graph embedding (Bose and Hamilton, 2019; Rahman et al, 2019). However, the fairness of HGNNs is still an unsolved problem, which is an important research direction in the future.

Robustness. Also, the robustness of HGNNs, especially the adversarial attacking, is always an important problem (Madry et al, 2017). Since many real-world applications are built based on HGs, the robustness of HGNNs becomes an urgent yet unsolved problem. What is the weakness of HGNNs and how to enhance it to improve the robustness need to be further studied.

Interpretability. Moreover, in some risk-aware scenarios, e.g., fraud detection (Hu et al, 2019b) and bio-medicine (Cao et al, 2020), 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 HGNNs. Besides, the emerging disentangled learning (Siddharth et al, 2017; Ma et al, 2019c), which divides the embedding into different latent spaces to improve the interpretability, can also be considered.

16.5.4 Applications

Many HG-based applications have stepped into the era of graph embedding. There have demonstrated the strong performance of HGNNs on E-commerce and cybersecurity. Exploring more capacity of HGNNs 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 HGs. Therefore, HGNNs are expected to open up broad prospects for these new areas and become a promising analytical tool. Another area is the biological system, 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 (Tsuyuzaki and Nikaido, 2017). HG structure has been applied to biological system as an analytical tool, implying that HGNNs are 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 (Zhao et al, 2019b). Therefore, successful technique deployment in various applications while resolving the scalability and efficiency challenges will be very promising.

Editor's Notes: The concept of the heterogeneous graph is essentially originated from the data mining domain. Although heterogeneous graphs can usually be formulated as attributed graphs (Chapter 4), the research focus of the former is typically the frequent combinatorial patterns of node types in a subgraph (usually a path). Heterogeneous graphs represent a wide range of real-world applications which usually consist of multiple, heterogeneous data sources. For example, in recommender systems introduced in Chapter 19, we have both the "user" node and "item" node as well as higher-order patterns formed by multi-node types. Similarly, molecules and proteins as well as many networks in Natural Language Processing and Program Analysis can also be considered as heterogeneous graphs (see Chapters 21,22,24,25).