Machine Learning (2023) 112:4227–4256
https://doi.org/10.1007/s10994-022-06160-5

Heterogeneous graph embedding with single-level aggregation and infomax encoding

Nuttapong Chairatanakul[1,3] · **Xin Liu**[2,3] · **Nguyen Thai Hoang**[1] · **Tsuyoshi Murata**[1,3]

Received: 1 March 2021 / Revised: 25 December 2021 / Accepted: 19 February 2022 /
Published online: 5 April 2022

Abstract
There has been an increasing interest in developing embedding methods for heterogeneous graph-structured data. The state-of-the-art approaches often adopt a bi-level aggregation scheme, where the first level aggregates information of neighbors belonging to the same type or group, and the second level employs the averaging or attention mechanism to aggregate the outputs of the first level. We find that bi-level aggregation may suffer from a down-weighting issue and overlook individual node information, especially when there is an imbalance in the number of different typed relations. We develop a new simple yet effective single-level aggregation scheme with infomax encoding, named HIME, for unsupervised heterogeneous graph embedding. Our single-level aggregation scheme performs relation-specific transformation to obtain homogeneous embeddings before aggregating information from multiple typed neighbors. Thus, it emphasizes each neighbor's *equal* contribution and does not suffer from the down-weighting issue. Extensive experiments demonstrate that HIME consistently outperforms the state-of-the-art approaches in link prediction, node classification, and node clustering tasks.

Keywords Heterogeneous information network · Graph neural network · Graph embedding · Infomax

1 Introduction

Graph structured data appear in many applications, including scientific discovery, social network analysis, web searching (Inokuchi et al., 2003), recommender systems, and geographical data (Miller & Han, 2001). Many techniques have been successfully developed to exploit both the information captured by the graph structure and the features of nodes and

Editors: Annalisa Appice, Grigorios Tsoumakas.

✉ Nuttapong Chairatanakul
nuttapong.c@net.c.titech.ac.jp

[1] Department of Computer Science, Tokyo Institute of Technology, Tokyo, Japan

[2] Artificial Intelligence Research Center, AIST, Tokyo, Japan

[3] AIST-Tokyo Tech Real World Big-Data Computation Open Innovation Laboratory, Tokyo, Japan

edges. Most notably, neural network approaches (Kipf & Welling, 2016; Hamilton et al., 2017; Veličković et al., 2017) and network embedding approaches (Perozzi et al., 2014; Tang et al., 2015b; Grover & Leskovec, 2016; Wang et al., 2016; Cao et al., 2015; Goyal & Ferrara, 2018) have continuously set the state of the art in a wide range of problems such as node classification, graph classification, and link prediction. However, these methods mentioned above often share a common homogeneity assumption. Thus, they are not suitable for graph data with various node types and edge types.

In real-world applications such as recommender systems or search engines, the graph data usually contain multiple types of objects (nodes) and relations (edges). Although it is reasonable to use a homogeneous graph learning model with node types and edge types (relations) encoded into node attributes, doing so would compromise the smoothness assumption inherent in many graph neural networks (NT & Maehara, 2019). In fact, node types (and also edge types) are discrete values and often do not share the same structure as node features. Therefore, *heterogeneous graphs* or in another name *heterogeneous information networks* (HINs) (Sun & Han, 2013), can capture data in real-world applications more truthfully than homogeneous graphs.

HINs are designed to capture rich semantics and comprehensive information. It is useful for various data mining tasks, such as similarity search (Sun et al., 2011), recommendation (Liu et al., 2014), clustering (Sun et al., 2012), and classification (Kong et al., 2012). The heterogeneity of HINs has posed a challenge in graph mining, that is how to learn information from multiple types of nodes and edges. Since the state of the art for homogeneous graph representation learning is neural-based graph embedding methods (Perozzi et al., 2014; Tang et al., 2015b; Grover & Leskovec, 2016; Wang et al., 2016; Cao et al., 2015; Kipf & Welling, 2016; Veličković et al., 2017), it is natural to extend these methods to HINs. Early works (Tang et al., 2015a; Dong et al., 2017) support multiple node types and relations, but their node embeddings do not consider *target relations*. To address this problem, some subsequent works (Ty et al., 2017; Shi et al., 2018b) implicitly consider the target relation as edges or metapath vectors. However, they do not consider neighbor information, which is crucial to the high performance as that in homogeneous graphs (Kipf & Welling, 2016; Hamilton et al., 2017).

Most recently, graph neural networks designed for HINs extend ideas from the homogeneous data literature to efficiently solve problems of heterogeneous data — setting state-of-the-art results . In general, these neural networks use an approach called *bi-level* aggregation (Fig. 1c) in order to learn the heterogeneous node embeddings. The first level aggregates information of neighbors with the same node types, relations, or meta-paths. Then, the second level employs the averaging or attention mechanism to aggregate the outputs of the first level. Notable bi-level aggregations include RGCN (Schlichtkrull et al., 2018), HAN (Wang et al., 2019), GATNE (Cen et al., 2019), and HGT (Hu et al., 2020b).

In this paper, we find that the bi-level approach may overlook individual node information, especially when there is an imbalance in the number of different typed relations. Take Fig. 1c as an example. Suppose there are many more *user interactions* than *also-buy* or *also-view* relations, then the bi-level aggregation scheme tends to down-weight the information from an individual user. This harms its performance for HIN embedding.

To further investigate this problem, we create a toy HIN, which consists of three types of nodes, namely "user", "item", and "tag" and two relations, namely "user-item" (U-I) and "item-tag" (I-T). To simulate *preference* of a user and *features* of an item, we randomly assign four and three tags to each user and item, respectively. Then, we connect an item and a tag if the item is associated with the tag; we add a link between a user and an item if they have two associated tags in common with more than 25% probability. As a result, the

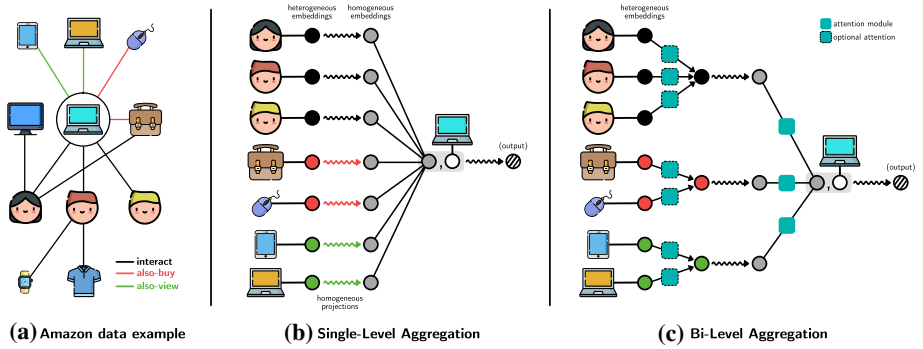


Fig. 1 **a** A snapshot of the Amazon data where the node of interest is in a circle (the laptop); **b** Example for our proposed the single-level aggregation scheme; **c** Example for the bi-level aggregation scheme where attention is required. Our model introduces relation projections and infomax learning, which replace the attention mechanism in state-of-the-art approaches

Table 1 Link prediction results (MRR) in the toy HIN

	U-I (96.4%)	I-T (3.6%)
GraphSAGE	54.4	78.9
GAT	54.4	80.8
RGCN	54.7	62.6
HAN	52.1	58.6
HGT	52.0	79.8
HIME	**55.6**	**89.4**

graph contains 1,000 users, 100 items and 10 tags and 8,025 U-I edges and 300 I-T edges, representing majority and minority relations, respectively. We generate 10 attributes of nodes according to their associated tags and another 20 non-related *noisy* attributes according to binary distribution. Table 1 shows the link prediction results of various methods (the experimental settings are described in the experiment section). Surprisingly, we find that bi-level aggregations (RGCN, HAN) are even inferior to traditional aggregations that do not consider heterogeneity (GraphSAGE (Hamilton et al., 2017) and GAT (Veličković et al., 2017)), especially for I-T relation. The reason is that bi-level aggregations down-weight U-I relation and get overfitted to the noisy features. More evidence, explanations, and results on the down-weighting issue will be provided in the experiment section and Sect. 4.3 **Bi-level vs Single-level Aggregation**.

In this paper, we propose a simple yet effective *single-level* aggregation scheme with infomax encoding, named HIME, for unsupervised HIN embedding (Fig. 1b). The key point is that we perform relation-specific transformation to obtain homogeneous embeddings before aggregating information from multiple typed neighbors. Thus, we emphasize the "equal" contribution of each neighbor and thus will not suffer from the down-weighting issue when there are imbalanced numbers of multiple typed relations. Our final embeddings are learned by a loss that encourages closeness between neighbors and an infomax encoder (Fig. 2) to augment graph smoothing in the homogeneous embeddings. As shown in Table 1, HIME outperforms the bi-level aggregation approaches, especially by a large

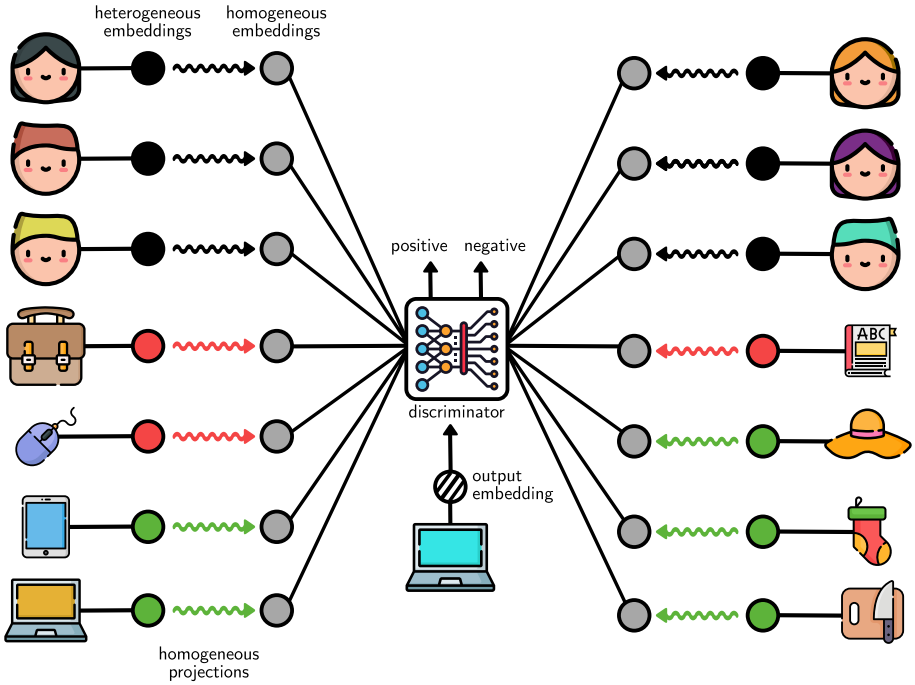


Fig. 2 Architecture of Heterogeneous Graph Infomax Encoding. Following Figure 1, nodes and relations on the left are neighbors with correct relations (positive) to the center node, while nodes and relations on the right are randomly sampled (negative)

margin for I-T relation, in the toy HIN. We do extensive experiments using ten benchmark datasets and demonstrate that HIME consistently outperforms the state-of-the-art approaches in link prediction, node classification, and node clustering tasks. We show that HIME is scalable and is able to deal with a large HIN containing 12.8 million edges.

Contribution—We make the following contributions: (1) To the best of our knowledge, we are the first to raise the down-weighting issue of bi-level aggregations hindering their effectiveness for HIN embedding and provide concretely empirical evidence. (2) We introduce the heterogeneous *single-level* aggregation scheme with infomax embedding, a simple yet effective HIN embedding method. (3) We show that our implementation outperforms the latest HIN embedding models in many practical tasks.

2 Preliminaries

Definition 1 (Heterogeneous Information Network) A Heterogeneous Information Network (HIN) is a tuple $(G, \mathcal{T}, \varphi, \mathcal{R})$, where $G = (\mathcal{V}, \mathcal{E})$ is an undirected graph with node set $\mathcal{V}$ and edge set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{R} \times \mathcal{V}$, and $\mathcal{R}$ is a set of relations. $\mathcal{T}$ is the set of node types and function $\varphi : \mathcal{V} \mapsto \mathcal{T}$ maps a node to a single type. Optionally, some dataset include node attributes $x' : \mathcal{V} \mapsto \mathbb{R}^p$, where p is the number of dimensions of the attributes. Note that HINs require $|\mathcal{T}| > 1$ or $|\mathcal{R}| > 1$.

Problem 1 (Relation-aware embedding for HIN) Given a HIN and a target relation $t \in \mathcal{R}$, the problem is to generate low-dimensional representation vectors $\mathbf{x}_i^t \in \mathbb{R}^d$ for each node $v_i \in \mathcal{V}$ according to the relation t and $d \ll |\mathcal{V}|$ such that the representations preserve the structure of the given HIN.

Our defined problem is common for unsupervised HIN embedding. To comprehensively embed HIN, we do not look at a specific relation to generate the embedding, but to solve the relation-aware embedding problem across all relations in a given HIN. The main challenge is maintaining a high-quality representation across relations while dealing with the scalability issue. In the following, we first discuss related works on graph embedding and HIN embedding.

2.1 Graph embedding

Recently, homogeneous graph embedding methods (Perozzi et al., 2014; Tang et al., 2015b; Grover & Leskovec, 2016; Wang et al., 2016; Cao et al., 2015) have emerged as scalable ways to learn low-dimensional vector representations for graph nodes. These node representations encode semantic information transcribed from the graph and can be used directly as the node features for downstream machine learning tasks. Traditionally, node representation can be obtained by performing the eigendecomposition of Laplacian matrix (Ng et al., 2002). However, due to the complexity of such operation, other heuristics were proposed. The most notable is DeepWalk (Perozzi et al., 2014), where we generate a series of random walks to capture the graph semantic and learn node representations using a skip-gram model (Mikolov et al., 2013). The applications of the skip-gram model can also be seen in recommender systems (Grbovic et al., 2015; Bianchi et al., 2020), context prediction (Lazaridou et al., 2015), etc. Subsequent models to DeepWalk include LINE (Tang et al., 2015b) and node2vec (Grover & Leskovec, 2016). Certain works (Wu et al., 2018; Yang et al., 2019) introduce hashing techniques to reduce the training time and improve the scalability. For a more comprehensive view, we refer to the survey article (Goyal & Ferrara, 2018) on graph embedding.

More recently, researchers proposed Graph Neural Networks (GNNs) as a new class of graph embedding models (Hamilton et al., 2017; Kipf & Welling, 2016; Veličković et al., 2017). While not learning the node embedding explicitly, these models implicitly learn the node embeddings by combining node attributes and graph structures in neural-based models. We refer to the graph neural network survey article (Wu et al., 2019) for more details in this literature.

Since real-world data such as the Amazon data (Fig. 1a) are intrinsically heterogeneous, it is not trivial to extend graph embedding methods to work with heterogeneous data. Many HIN embedding models have been successful in bridging this gap.

2.2 HIN embedding

In early works of mining HINs, many methods (Sun et al., 2011; Liu et al., 2014) use meta-paths as semantic information to underline the difference between HIN and homogeneous network. As random-walk based models become popular, there are many attempts to combine the concept of meta-paths and skip-gram models. The notable works are metapath2vec and HIN2vec. metapath2vec (Dong et al., 2017) formalizes meta-path-based random walk and then utilizes heterogeneous skip-gram models. HIN2vec considers a neural network

model for capturing and differentiating the information of meta-paths. In the meantime, another line of work (Xu et al., 2017; Tang et al., 2015a; Shi et al., 2018a) treat the graph under each relation as a *subgraph*, then they jointly learn from those subgraphs.

Researchers have recognized the problem of incompatibility of heterogeneity in relations (Shi et al., 2018b; Chen et al., 2018). To alleviate this problem, they propose relation-specific projection or implicitly consider it as relation embedding. With the success of GNNs, many works follow these architectures and adapt them to HINs. HAN (Wang et al., 2019), GATNE (Cen et al., 2019) and GTN (Yun et al., 2019) utilize attention mechanism after aggregating on *subgraph-level* separated by node types, relations, or meta-paths. We refer to these methods as *bi-level aggregations*. However, none of these concerns about the contribution of each individual neighbor to the final node representations.

3 Proposed method

We present a simple yet effective relation-aware heterogeneous graph embedding algorithm called **H**eteregeneous graph **I**nfo**M**ax **E**ncoder (HIME) to keep the awareness of relations in heterogeneous graphs without suffering from the down-weighting issue. We first describe our encoder to learn node representations, then we introduce a mechanism to maximize the mutual information inside the encoder.

3.1 Relation-aware node embedding

Initially, each node v_i is associated with a feature vector $\mathbf{x}_i^{(0)} \in \mathbb{R}^d$ which is shared across relations. In case node attributes are available, we project them to a d-dimensional space by $\mathbf{x}_i^{(0)} = \mathbf{W}_a \mathbf{x}'_i$ where $\mathbf{W}_a \in \mathbb{R}^{d \times p}$ is a learnable weight matrix for the projection and $\mathbf{x}'_i \in \mathbb{R}^p$ denotes attribute of v_i. To get the final representation of node v_i for relation t, we combine the information of node v_i, its neighbors $\mathcal{N}_i$, and relation t together. The process is fulfilled by a single or a stack of our proposed *heterogeneous single-level* aggregator(s).

3.1.1 Heterogeneous single-level aggregator

As shown in Fig. 1b, given an object of interest, we transform its neighbor features according to their relations to v_i, $\mathbf{e}_{j,r} = h(\mathbf{x}_j, r)$ where v_j is a node connected to v_i by relation r; $h : \mathbb{R}^d \times \mathcal{R} \rightarrow \mathbb{R}^d$ is a differentiable function. h allows a node to pass distinct feature for different relations. The transformation can be a hyperplane translation (Wang et al., 2014) or linear transformation. We call $\mathbf{e}_{\cdot,\cdot}$ as an edge vector. Then we aggregate edge vectors using the mean function: $\mathbf{n}_i = \frac{1}{|\mathcal{N}_i|} \sum_{(j,r) \in \mathcal{N}^t} \mathbf{e}_{j,r}$. Notice that we perform the neighbor relation-specific transformation before the aggregation to emphasize each neighbor's contribution rather than diluting the node's information by fusing it with neighbors of the same relation. Furthermore, we found that the transformation should be carefully selected. For example, the hyperplane translation assumes that all edge vectors after the transformation still belong to the same feature space but in different hyperplanes. This can be beneficial for certain heterogeneous graphs, where the information between relations is highly inclusive so that knowing other relations can help to inform the structure of a target relation.

Next, we combine the information of node v_i ($\mathbf{x}_i$) and its neighbors ($\mathbf{n}_i$) by using gated recurrent units (GRU) (Cho et al., 2014): $\mathbf{x}_i \leftarrow \text{GRU}(\mathbf{x}_i, \mathbf{n}_i)$. We treat $\mathbf{x}_i$ as the *hidden state* of a recurrent model. Alternatively, simplified versions (Chairatanakul et al., 2019) of GRU can be used to reduce the model complexity and possibly improve the performance in sparse datasets. The motivation behind using GRUs is based on the oversmoothing issue of GNNs; That is the degradation of performance of GNNs when increasing the number of layers because of the similarity in node representations (Li et al., 2018; Oono & Suzuki, 2019). Note that GRU is commonly used in learning from sequential data that retaining past information is crucial, although it requires heavy computational resources. We hypothesize that GRU can extract new information of the current layer while retaining useful information from the previous layer. Any homogeneous GNN can also be used after obtaining edge vectors by treating them as neighbors in a homogeneous graph. Note that a slight modification of bi-level aggregation can turn it into single-level aggregation. However, doing that will contradict the purpose of the second-level that aims to reassign the weight based on relations.

In the last layer, we employ a distinct GRU for each target relation t. The formula can be written as follows: $\mathbf{x}_i^t = \text{GRU}_t(\mathbf{x}_i, \mathbf{n}_i)$. The motivation is to allow the node representation to have both shared information across all relations and unique node's relational information. This can be realized by looking at updating node vector of GRU_t: $\mathbf{x}_i^t = (1 - \mathbf{z}_i^t) \cdot \mathbf{x}_i + (\mathbf{z}_i^t) \cdot \hat{\mathbf{x}}_i^t$, where z_i^t is the update gate vector and $\hat{\mathbf{x}}_i^t$ is the relation-specific *candidate* vector.

3.1.2 Objective function

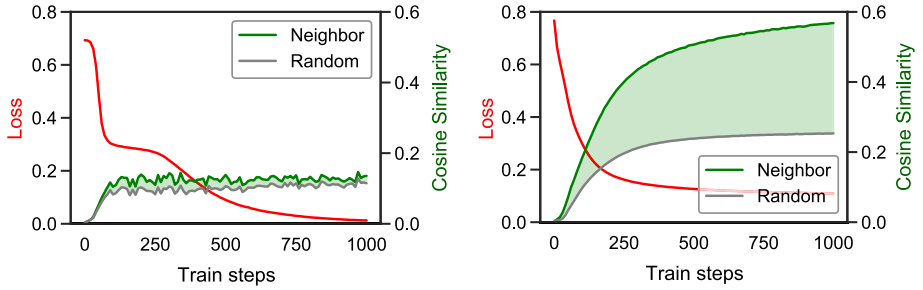
To preserve the structure of HIN, we encourage the closeness of embeddings for nodes connected by an edge, while enforcing the separation of embeddings for nodes unconnected. Therefore, we minimize the following loss:

$$\mathcal{L}_G = \frac{1}{|\mathcal{E}|} \sum_{v_i \in \mathcal{V}} \sum_{(j,t) \in \mathcal{N}_i} \sum_{(k,t) \notin \mathcal{N}_i} -\log\, \sigma(\langle \mathbf{x}_i^t, \mathbf{x}_j^t \rangle - \langle \mathbf{x}_i^t, \mathbf{x}_k^t \rangle), \tag{1}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product. The loss is derived from *Bayesian Personalized Ranking* (Rendle et al., 2009). It is commonly used in recommender systems (Ricci et al., 2010) and is similar to margin-based ranking loss (Bordes et al., 2013). For scalability, we use negative sampling, which will be informed later in Sect. 3.5.

3.2 Heterogeneous graph infomax encoding

While the proposed aggregator in the previous section is flexible and powerful in learning local structures, using multiple transformations can result in high heterogeneity. That is, the edge vectors conflict with each other. This impairs the graph *smoothing* which is an essential characteristic of GNNs (Chen et al., 2020; Xie et al., 2020). We want to avoid this scenario and, at the same time, encourage the model to capture unique local features. Therefore, we aim to maximize the mutual information between edge vectors and the output of the layer. Similar to DGI (Veličković et al., 2018) and DIM (Hjelm et al., 2018), we use the binary cross-entropy between samples from the joint (positive) and the product of marginals (negative):



(a) Global Infomax **(b)** Infomax Encoding

Fig. 3 Average of the cosine similarity of edge vectors between neighbors of the same node (*green line*) and between any random edges (*grey line*). The green area shows the gap between them. The larger the gap, the better homogeneity and lesser prone to oversmoothing

$$\mathcal{L}_{\mathcal{I}}^{(l)} = \frac{1}{E_{\mathcal{I}}} \sum_{v_i \in \mathcal{V}} \Big(\sum_{(j,r) \in \mathcal{N}_i} -\log \mathcal{D}^{(l)}(\mathbf{e}_{j,r}^{(l-1)}, \mathbf{x}_i^{(l)}) + \sum_{(k,q) \in \tilde{\mathcal{N}}_i} -\log\,(1 - \mathcal{D}^{(l)}(\mathbf{e}_{k,q}^{(l-1)}, \mathbf{x}_i^{(l)})) \Big), \quad (2)$$

where $E_{\mathcal{I}} = \sum_{v_i \in V}(|\mathcal{N}_i| + |\tilde{\mathcal{N}}_i|)$, superscript (l) denotes l-th layer of the aggregator, $\mathcal{D}^{(l)} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a discriminator for l-th layer and $\tilde{\mathcal{N}}_i$ is a corrupted neighbors of node v_i. As analogous to DIM, the vicinity centering around v_i in the graph is treated as a single *image*. To increase the mutual information, the encoder needs to consider every neighbors and aggregates the information in which most neighbors *agree*. Notice that for each layer, we apply the loss separately since it is easier to apply mini-batch training.

Following DGI, we adopt its discriminator: $\mathcal{D}(\mathbf{x}, \mathbf{y}) = \mathbf{x}^{\intercal}\mathbf{U}\mathbf{y}$, where $\mathbf{U}$ is a weight matrix of the discriminator. We call the process of optimizing Eq. (2) via a discriminator inside each layer of the single-level aggregator as *infomax encoding*. Note that our motivation for the infomax encoding is different from the graph infomax in DGI, DMGI (Park et al., 2020), and HDGT (Ren et al., 2019). In their works, the main objective is to preserve the mutual information between *local patches* and *global graph summary*. We will call them as *global infomax*.

The advantages of the proposed infomax encoding over global infomax are two-fold. First, infomax encoding is compatible with mini-batch sampling, whereas global infomax needs to compute the embeddings for the whole graph to obtain the global summary. This makes infomax encoding scalable to a large graph without any further modification. The second is for promoting homogeneity between neighbors or edge vectors in the graph. To make a comparison, we optimize the model in the previous section via either infomax encoding or global infomax[1] while measuring the homogeneity between neighbors. To measure the homogeneity, we use the cosine similarity between neighbors of the same node and between any random edges following Chen et al., (2020). Intuitively, we want the high value from the neighbors telling that the neighbors contain similar information, while we want the low value from the random indicating oversmoothing. The results are plotted in Fig. 3. we can clearly see the large gap between the value from the neighbors and the value from the random in Fig. 3b. This indicates that infomax encoding can encourage the homogeneity while keeping the oversmoothing in control. On the other hand, Fig. 3a

[1] We choose DGI to represent global infomax because it is the most well-known and easy to implement.

informs that global infomax does not deliver the same desirable. More benefits of infomax encoding will be presented in Sect. 4.4 **Study of Infomax Encoding**.

3.3 Model Optimization

For the final objective, we jointly optimize the loss from the graph context in Eq. (3) and the *infomax* loss in Eq. (2):

$$\mathcal{L} = \mathcal{L}_G + \alpha \sum_{l=1}^{L} \mathcal{L}_{\mathcal{I}}^{(l)}, \tag{3}$$

where α is a hyper-parameter controlling the importance of the infomax loss. Since it is computationally expensive to minimize the above loss directly, we adopt the negative sampling technique. Specifically, we uniformly sample an edge (v_i, t, v_j) from the graph as a positive sample, then we uniformly sample K negative nodes that have the same types as v_j and do not have relation t to v_i. For further improvement on the negative sampling, "hard-samples" negative sampling (Zhu et al., 2021) or adversarial negative sampling (Hu et al., 2019; Sun et al., 2019) can be considered. The effect of negative sampling in homogeneous graphs have also been investigated by Qiu et al., (2018); Yang et al., (2020).

Note that the model can perform in semi-supervised manner by changing the loss function $\mathcal{L}_G$ to a loss function for multiclass classification such as cross-entropy loss. However, we found that the semi-supervised loss usually converges significantly faster than the infomax loss. The slower convergence of the infomax loss suggests that the model may underutilize the infomax encoder in such a condition.

To make the model applicable to large graphs, we follow the architecture of GraphSAGE (Hamilton et al., 2017) to be able to generate node representation individually and utilize mini-batch training. In particular, for generating the representation of a node, namely v_i, we uniformly sample up to n neighbors of v_i, where n is a hyper-parameter. Subsequently, for each sampled neighbor, we perform the sampling process of that neighbor. We repeat the process for L times. The aggregation considers only these samples and v_i for generating the representation of v_i. In this way, we can limit the memory usage in the aggregation process to $O(n^L T)$, where T is the space complexity of the transformation h, compared with $O(|\mathcal{E}|T)$ using the whole graph. In practice, L is usually set to a small number due to the oversmoothing effect. We obtain nodes' corrupted neighbors by shuffling the nodes' neighbors of the same mini-batch. Specifically, $\tilde{\mathcal{N}}_{B_i} = \mathcal{N}_{B_{\pi(i)}}$ such that $T_i = T_{\pi(i)}$ where π is a permutation function, B and T denote the arrays of target relations and node indexes of a mini-batch, respectively. The reason is to reduce computation cost. Since we need to perform $l-1$ encoding layers to has $\mathbf{e}_{k,q}^{(l-1)}$ in Eq. (2), by shuffling the neighbors inside a mini-batch, $\mathbf{e}_{k,q}^{(l-1)}$ has been calculated already in the positive side and can be reused in the negative.

4 Experiments

We conduct extensive experiments to compare HIME with the state-of-the-art models on link prediction, node clustering, and node classification tasks. We also analyze the benefits of the proposed single-level aggregation over bi-level aggregation. Moreover, we provide a thorough investigation into the different aspects of HIME, including the effect of

Table 2 Statistics of datasets

Dataset	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{R}\rvert$
DBLP	35,796	168,703	4	3
Yelp	30,651	331,490	4	4
Douban Movie (DM)	37,441	1,686,025	5	5
Douban Book (DB)	41,959	2,130,693	7	7
Amazon-Large	1,025,457	12,769,007	2	3
Amazon	10,166	148,865	1	2
YouTube	2,000	1,310,617	1	5
Twitter	10,000	331,899	1	4
ACM	3,025	2,240,042	1	2
IMDB	3,550	80,216	1	2

infomax encoding, the scalability, hyper-parameter sensitivity, and its adaptability to other frameworks.

4.1 Datasets

Aside from our synthetic toy dataset, we use ten publicly available real-world HIN datasets. We divide the datasets into three groups. The first group is datasets where both node types and relations are more than one: DBLP (Hu et al., 2019), Yelp (Hu et al., 2019), Douban Movie (Shi et al., 2019), Douban Book (Shi et al., 2019), and Amazon-Large[2] (Ni et al., 2019). The second group is multiplex networks ($|\mathcal{T}| = 1$ and $|\mathcal{R}| > 1$): Amazon, YouTube, and Twitter. We obtain the data from Cen et al., (2019). The last group is multiplex networks with node labels: ACM and IMDB. We obtain node labels and attributes from Park et al., (2020) for node clustering and classification evaluation purpose. Basic statistics of the datasets are summarized in Table 2. Additional details can be found in Sect. A in the appendix.

4.2 Compare HIME with baselines

We first compare our proposed HIME to various unsupervised models which can be potentially applied to HINs:

- Traditional homogeneous graph embedding: LINE (Tang et al., 2015b), DeepWalk (Perozzi et al., 2014)[3]
- HIN-based embedding: BHIN2vec (Lee et al., 2019b), metapath2vec (Dong et al., 2017), HEER (Shi et al., 2018b)
- Knowledge graph embedding: TransE (Bordes et al., 2013), DistMult (Yang et al., 2014), CompleEx (Trouillon et al., 2016), RotatE (Sun et al., 2019)

[2] We use the data in "sports and outdoors" category.

[3] Note that DeepWalk can serve as a representative of skip-gram (Mikolov et al., 2013) based methods, which have been successfully applied to different problems such as Prod2vec (Grbovic et al., 2015; Bianchi et al., 2020) for recommender systems and multimodal models (Lazaridou et al., 2015) for context prediction.

- GNN for homogeneous graph: GraphSAGE (Hamilton et al., 2017), DGI (Veličković et al., 2018)
- GNN for multiplex network: DMGI (Park et al., 2020)
- GNN for HIN: RGCN (Schlichtkrull et al., 2018), HAN[4] (Wang et al., 2019), GATNE (Cen et al., 2019), and HGT (Hu et al., 2020b).

Note that all GNNs for HIN in the list are bi-level aggregations. Because the core idea of DMGI (Park et al., 2020) and HDGI (Ren et al., 2019) is similar but DMGI has an improved regularization, we select DMGI to represent global infomax approach for multiplex networks. GATNE in Link Prediction section refers to GATNE-T while GATNE in Node Clustering and Classification section refers to GATNE-I. GATNE* refers to a variant of GATNE by changing its initialization and loss function to Eq. (1). HAN* refers to a variant of HAN by changing the semantic attention to be an independent and learnable parameter. For a fair comparison, we fix the embedding dimensions d to 128. Without specifying, all models are trained in unsupervised manner. Please see Sect. B in the appendix for additional details about the implementation and hyper-parameter settings.

4.2.1 Link prediction

To evaluate the quality of embedding methods on preserving the information of a HIN, we conduct link prediction experiments following Shi et al., (2018b). For each HIN, we split its edges into three sets for training, validation, and testing with numbers 85, 5, and 10% from the total, respectively. An evaluated model is trained on the training set, while the validation set is used for stopping criteria and finding suitable parameters. The task is to predict the edges in the test set using the learned embeddings from the training set. We sample 10 negatives instead of all possible candidates because of computation cost. We rank positive edges among both positives and negatives and report the mean reciprocal rank (MRR) by micro- and macro-average. In particular, the micro-average MRR averages all reciprocal rank without considering relations, whereas macro-average MRR averages over the mean of values of reciprocal rank associated with each relation. Mathematically,

$$\text{MRR}_{micro} = \frac{1}{|\mathcal{E}_{test}|} \sum_{(i,t,j)\in\mathcal{E}_{test}} \frac{1}{2}\left(\frac{1}{rank_i} + \frac{1}{rank_j}\right),$$

$$\text{MRR}_{macro} = \frac{1}{|\mathcal{R}|} \sum_{t\in\mathcal{R}} \frac{1}{c^t} \sum_{\substack{(i,r,j)\in\mathcal{E}_{test} \\ r=t}} \frac{1}{2}\left(\frac{1}{rank_i} + \frac{1}{rank_j}\right),$$

where $\mathcal{E}_{test}$ denotes the test set and c^t denotes the number of test edges with a relation t in the test set. We average over relations for which their appearances exceed 5%. Table 3 lists the results. We observe that HIME achieves the best performance for all cases. Particularly, HIME significantly outperforms the baselines, with p-value < 0.01.

[4] HAN was proposed for semi-supervised learning. For training in unsupervised manner, we apply RGCN training framework.

Table 3 Micro- and macro-average MRR results of each model in link prediction task in HINs. Bold texts indicate the best result in each case. Underline texts indicate the best among baselines. ⋆ indicates the significant improvement of HIME over baselines, with p-value < 0.01

	DBLP		Yelp		Douban Movie		Douban Book	
	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Deepwalk	57.46	50.56	54.43	49.88	35.26	35.86	44.51	42.65
LINE-1st	59.25	56.67	68.60	62.96	54.74	55.63	61.07	59.94
LINE-2nd	37.51	42.40	44.92	46.33	44.77	49.15	33.40	32.65
metapath2vec	64.26	62.77	72.59	70.94	61.16	63.99	55.66	53.72
BHIN2vec	65.66	67.31	72.89	68.96	58.66	61.34	55.28	53.36
HEER	70.82	65.64	76.12	71.07	69.01	71.48	66.17	65.11
TransE	54.96	55.98	68.56	68.21	53.63	56.81	44.54	43.98
DistMult	61.11	58.33	75.63	73.61	69.48	72.49	65.20	63.96
ComplEx	59.47	57.15	71.55	70.29	68.17	71.23	64.01	62.72
RotatE	65.72	66.54	76.65	74.16	67.72	70.61	61.20	60.44
GraphSAGE	63.14	57.60	68.07	65.08	49.04	51.28	50.28	49.05
RGCN	65.29	66.16	74.06	70.56	60.20	62.85	58.74	57.85
HAN	51.23	53.09	50.95	45.83	44.53	47.67	37.78	36.61
HAN*	66.47	66.63	76.68	72.41	61.20	64.32	57.68	56.46
HGT	62.92	64.72	74.07	70.89	61.04	64.20	57.02	55.69
GATNE	45.14	49.66	71.94	66.79	48.72	51.25	42.22	42.24
GATNE*	60.47	62.44	79.24	74.80	69.84	72.44	65.55	64.49
HIME	**72.88**⋆	**69.83**⋆	**80.05**⋆	**76.31**⋆	**73.29**⋆	**75.56**⋆	**68.41**⋆	**67.41**⋆

Table 4 Performance for link prediction in multiplex networks. The result of [‡] are taken from the corresponding paper

	Amazon		YouTube		Twitter	
	AUC	F1	AUC	F1	AUC	F1
TransE	88.77	82.61	78.45	70.74	63.30	61.09
DistMult	98.86	95.60	93.67	86.19	92.65	85.08
ComplEx	85.82	82.45	79.92	72.19	85.01	76.44
RotatE	98.05	93.66	84.95	75.64	93.15	86.22
DGI	88.27	80.36	77.95	71.18	91.52	84.06
DMGI	90.43	83.21	88.77	81.22	89.70	83.60
RGCN	98.05	93.72	85.47	77.50	91.92	84.31
HAN	95.59	90.00	63.59	60.25	86.02	81.27
HAN*	98.24	94.04	87.67	80.01	92.10	84.51
HGT	98.33	94.10	85.94	78.07	91.28	83.73
GATNE [‡]	97.44	92.87	84.61	76.83	92.30	84.96
GATNE*	98.11	94.04	92.43	84.76	87.35	79.41
HIME	**99.09**	**95.92**	**95.86**	**89.07**	**94.59**	**87.69**

Table 5 Node classfication results (MiF1, MaF1) and node clustering results (NMI) on multiplex networks. [†] refers to semi-supervised training. Bold texts indicate the best result in each case. Underline texts indicate the best among baselines

	ACM			IMDB		
	MaF1	MiF1	NMI	MaF1	MiF1	NMI
HEER	73.8	73.1	.433	48.8	51.0	.009
TransE	75.3	74.8	.390	45.7	50.2	.027
DistMult	69.9	69.1	.389	40.6	41.7	.009
ComplEx	77.4	76.6	.389	41.6	42.5	.005
RotatE	78.2	77.2	.389	43.4	44.0	.016
DGI	79.1	80.2	.644	59.3	60.7	.174
DMGI	88.5	88.7	.687	**63.9**	64.4	.193
RGCN	88.2	88.2	.654	62.2	61.4	.194
HAN	77.9	78.4	.510	60.1	59.0	.045
HAN [†]	88.2	88.3	.650	62.9	63.0	.163
HGT	88.0	88.0	.558	61.7	60.6	.196
GATNE*	66.5	66.0	.009	38.8	40.6	.020
HIME	**89.2**	**89.1**	**.721**	**63.9**	**64.6**	**.202**

4.2.2 Link prediction on multiplex networks

In multiplex networks, nodes of the same type are connected to each other with multiple relations. We conduct link prediction in multiplex networks to see whether our model can handle effectively this scenario. We obtain data from Cen et al., (2019) and conduct experiments following them. We use their source code for evaluation and report the performance.

We report a summary of the results in Table 4. We observe that HIME shows superior performance over other models. The highest gap in performance is in the YouTube dataset, which has the highest number of relations among those three datasets. This implies the effectiveness and distinguishability of the model for dealing with multiple relations. Although DMGI and DGI aiming to capture global properties are good for node clustering and classification, they are inferior to models with graph context optimization (first- and second-order in graphs) for the link prediction task.

4.2.3 Node clustering and classification

Node clustering aims to group nodes belonging to the same class, while node classification is to identify their classes which can be considered as either multi-class classification or multi-label classification (Tsoumakas and Katakis, 2007; Do et al., 2019). We conduct experiments for both cases following Park et al., (2020) with the same data split for training, validation, and test sets. For methods that ignore node attributes, we concatenate the raw attributes with the learned node representations following Park et al., (2020). Since our methods generate multiple node representations for different edge types, for a fair comparison, we select the node representations of an edge type that yields the best results in the validation set, then evaluate the performance on the test set. In practice, we can combine such different node representations and use fast and highly scalable feature selection methods including ensemble techniques to enhance the performance. We report normalized

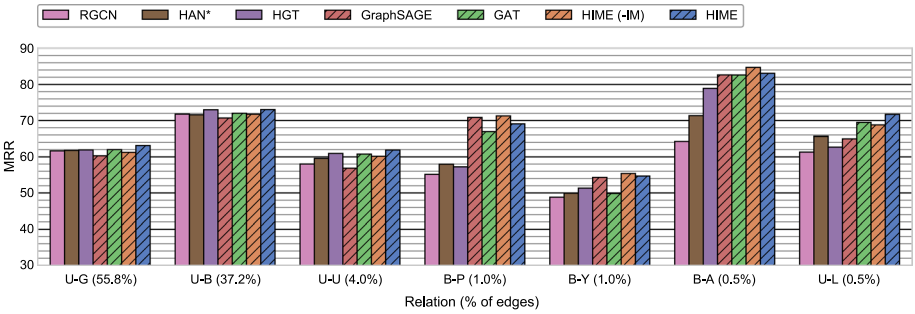


Fig. 4 Link prediction performance of models for each relation in Douban Book dataset by optimizing relation-aware embedding. The number in round brackets indicates the percentage of the relation

mutual information (NMI) for node clustering[5], and macro- (MaF1) and micro-average F1 (MiF1) for node classification. For comparison, we include bi-level aggregations: RGCN, HAN, and HGT trained in unsupervised manner with Eq. (1) as the objective function.

The results are summarized in Table 5. HIME demonstrates the best performance, while DMGI is the second best. For node classification, we observe that our unsupervised model HIME even outperforms the semi-supervised model HAN [†]. For node clustering in ACM dataset, HIME achieves performance gain as high as 4.9% over DMGI which is significantly superior to other baselines.

4.3 Bi-level vs single-level aggregation

In this section, we explain why bi-level aggregation performs worse than single-level aggregation for relation-aware HIN embedding. First, we look at where the performance is improved by single-level over bi-level aggregation. We conduct further experiments for the link prediction task on Douban Book dataset, which has the largest number of relations by switching between graph convolutional methods using the same objective function as defined in Sect. 3.1. For bi-level aggregations, we include well-known methods such as RGCN (Schlichtkrull et al., 2018), HAN (Wang et al., 2019), and HGT (Hu et al., 2020b), and for single-level aggregations, we include commonly used methods such as GraphSAGE (Hamilton et al., 2017) and GAT (Veličković et al., 2017) and also our proposed HIME. We also include our proposed single-level aggregator without infomax encoding as "HIME (-IM)".

Figure 4 presents the link prediction results for each relation. We can see that the single-level aggregations significantly outperform bi-level aggregations in minority relations. To explain why bi-level aggregations perform worse, we aim to investigate the importance of each relation to node representations for a minority relation. Intuitively, an appropriate aggregator should transfer abundant information from majority relations to the minority. One can simply look at the attention score of the bi-level. However, it lacks the consideration of the magnitude of a feature and the transformation along the aggregation path.

To provide a better concrete analysis, we derive an idea from a neural network pruning technique *SNIP* (Lee et al., 2019a) which can also be considered as an application of an attribution method *gradient*input* (Shrikumar et al., 2017). SNIP uses the derivative

[5] Node clustering is conducted on nodes in the test set, following Wang et al., (2019); Park et al., (2020).

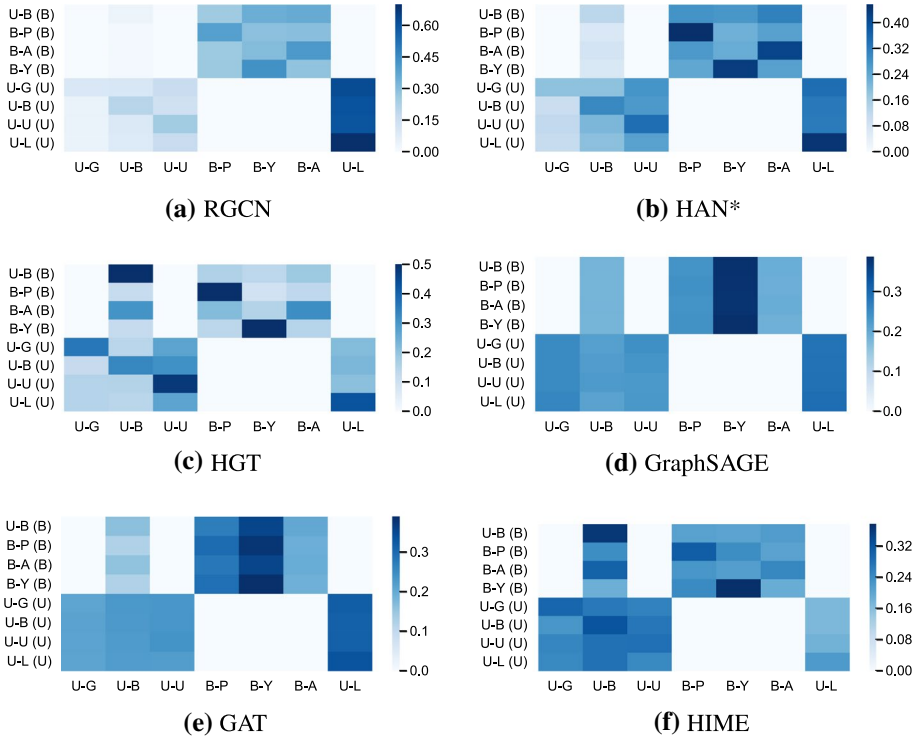


Fig. 5 Average saliency score of nodes connected to output nodes via each relation (x-axis) for generating the node representations for a target relation (y-axis) on Douban Book dataset

of a loss function with respect to an auxiliary variable c representing connectivity of a parameter w. The purpose is to find important connections based on the change of c called *saliency score*. To measure the importance of a node v_i, the saliency score of the node s_i can be calculated as:

$$s_i = \text{abs}\left(\frac{\partial \mathcal{L}_G(\mathbf{c}_i \odot \mathbf{x}_i^{(0)}; \theta, G)}{\partial \mathbf{c}_i}\Big|_{\mathbf{c}_i=\mathbf{1}}\right)^{\intercal}\mathbf{1}, \tag{4}$$

where θ is a set of the model's parameters, $\mathbf{x}_i^{(0)}$ denotes an *initial* feature vector of node v_i before the aggregation, and abs denotes a function calculating absolute value of each element in an input. The higher saliency score, the higher significance of a node.

Figure 5 shows the average nodes' saliency score of each relation to a target relation's node representations. Note that Fig. 5a—c are from bi-level aggregations, while Fig. 5d–f are from single-level aggregations. RGCN has very high saliency scores on minority relations because it uses the mean across relations in the second-level. HAN* uses attention mechanism in the second-level that can slightly gravitate the significance to majority relations and improve the performance. HGT deploys an attention mechanism considering each message from a neighbor instead. This can alleviate the down-weighting problem and make its results closer to those of single-level aggregations than others. However, we observe that for most target relations, HGT has very high saliency

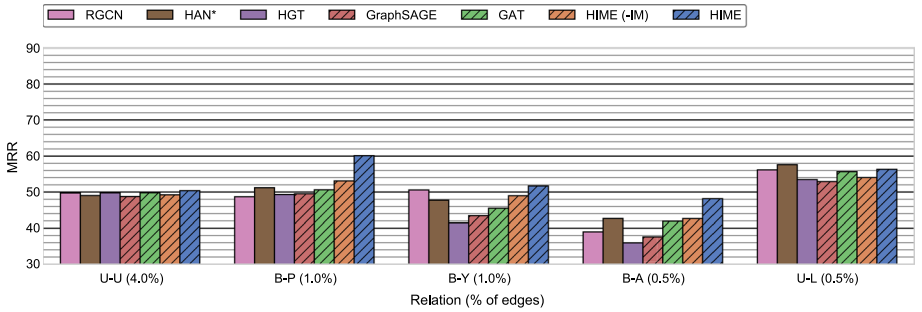


Fig. 6 Link prediction performance of models for each relation in Douban Book dataset by optimizing link prediction task for each relation. The number in round brackets indicates the percentage of the relation

scores on the same relation. This suggests that it tends to underutilize the information of HIN. However, the saliency scores of majority relations,"U-G" and "U-U", of bi-level aggregations are still much lower than those of minority relations for most cases. This demonstrates that bi-level aggregation scheme down-weights the information of the majority. On the other hand, Fig. 5d–f support that single-level aggregations much less suffer from the down-weighting issue.

However, the disadvantage of both GraphSAGE and GAT is the lack of awareness of relations in the aggregation that reduces their powerfulness when performing on HIN. For example, if we aim to directly optimize link prediction for each relation instead of relation-aware embedding. Figure 6 informs us that bi-level aggregation is better than single-level aggregation (excluding HIME) in all cases, following a similar finding by other researchers. This suggests that there is room for improvement to find a method that considers relations in the aggregation without suffering from the down-weighting issue. By considering relations in the aggregation to improve the powerfulness alone, we can see the increase in performance in HIME (-IM), while still performing well for minority relations in Fig. 4, unlike bi-level aggregations. Finally, by incorporating infomax encoding to encourage graph smoothing, we observe that HIME outperforms HIME (-IM) in all cases in Fig. 6 and most cases in Fig. 4 implying that it can utilize more information from graph structure. We can conclude that HIME is effective and powerful for HIN embedding and does not suffer from the down-weighting issue, satisfying both requirements.

4.4 Study of infomax encoding

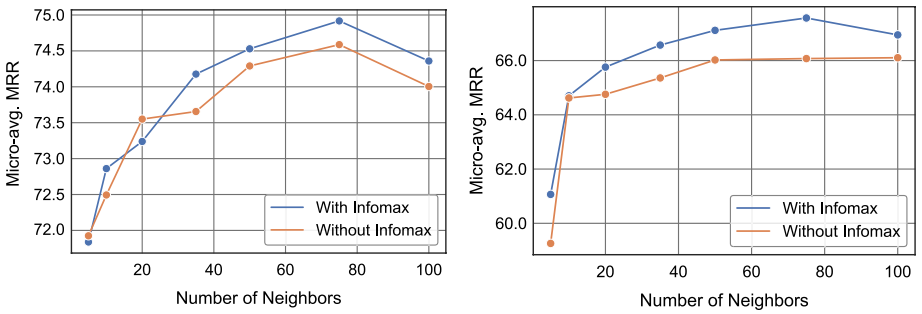
In this section, we provide study and analyze of the effect of the infomax encoding. For these purposes, we conduct further experiments with three settings: insusceptibility to attack on node features, horizontal improvement, and vertical improvement.

4.4.1 Insusceptibility to attack on node features

As we introduce the infomax encoding, we hypothesize that the infomax encoding augments the graph smoothness. Feng et al., (2019) show that implementing the graph smoothness in the prediction stage can increase the performance and robustness against attacks on node input features in node classification, and Jin & Zhang (2019) obtain a

Table 6 Node classification performance of HIME on IMDB graph with node feature attacks. Bold values indicate the best result in each case

Attack		HIME (-IM)		HIME	
Type	Node	MaF1	MiF1	MaF1	MiF1
Noise	Random	61.9	63.0	**64.1**	**64.7**
	Top	61.7	62.4	**63.3**	**64.0**
Shuffle	Random	61.4	62.2	**63.2**	**63.8**
	Top	60.0	61.4	**63.3**	**63.7**
Zero	Random	61.2	62.1	**63.7**	**64.0**
	Top	60.7	61.7	**63.7**	**64.2**
No (Original)		61.6	62.8	**64.7**	**65.1**



(a) Douban Movie **(b)** Douban Book

Fig. 7 Performance of HIME with/without infomax w.r.t. the number of sampled neighbors

similar achievement by perturbing the latent presentation in GCN. We will demonstrate that the infomax encoding follows the same conjecture and improves the robustness.

We choose the node classification task as in Sect. 4.2.3. We perform node feature attacks on either 5% uniformly sampled nodes or top 5% of nodes ranked by node degree. The attacks are 1. *Noise*: injecting Gaussian noise $\mathcal{N}(0, 0.01)$ 2. *Shuffle*: row shuffling between the attacked nodes 3. *Zero*: substituting constant for its features. Then we train HIME and evaluate the performance on an attacked graph. The results are collected and presented in Table 6. We can see that HIME with infomax encoding outperforms its variance without infomax encoding (-IM) regardless of attack types and attacked nodes.

4.4.2 Horizontal improvement

The second setting is to see its benefit when going *wide* (horizontal) in the graph. We set the number of layers L to 1, gradually increase the number of sampled neighbors N from 5 to 100, and run the model with and without the infomax encoding. We plot the performance of the model against the number of sampled neighbors in Fig. 7. We can see that the model achieves higher performance with infomax encoding than without it when N is high enough. This follows our assumption that infomax encoding can improve the usage of each neighbor's information.

Table 7 Performance (Micro-avg. MRR) of the model with and without infomax encoding by varying the number of layers

Dataset	Infomax	$L = 1$	$L = 2$	$L = 3$	$L = 4$	$L = 5$
DBLP	w/o	65.13	65.54	66.97	66.81	66.25
	w	**69.83**	**68.99**	**68.36**	**68.75**	**66.82**
Yelp	w/o	75.26	75.36	75.00	75.52	75.30
	w	**76.31**	**76.09**	**75.65**	**76.00**	**75.45**

Table 8 Performance (Micro-avg. MRR) of the model with different aggregation styles by varying the number of layers. Bold values indicate the best result in each case

Dataset	Style	$L = 2$	$L = 3$	$L = 4$	$L = 5$
DBLP	GCN	69.87	69.30	68.88	65.34
	RGCN	70.08	69.55	69.04	67.21
	GRU	**70.73**	**69.95**	**69.26**	**67.90**
Yelp	GCN	76.02	75.57	75.71	74.25
	RGCN	**76.25**	75.44	75.46	73.85
	GRU	76.04	**75.94**	**76.13**	**75.87**

4.4.3 Vertical improvement

In the last setting, we investigate whether the infomax encoding can improve the performance when going *deep* (vertical) in the graph. We conduct additional experiments by varying the number of layers L from 1 to 5, which can reach up to five-hop neighbors, and run the model with and without using infomax encoding in all layers. Due to the limitation in our memory space, we set the number of neighbors to 50, 20, 8, 5, 3 for L from 1 to 5.

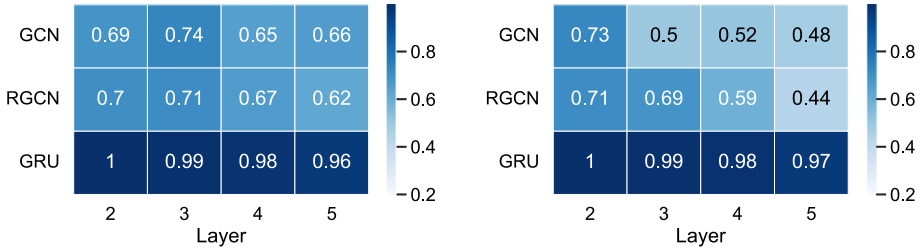
The results are listed in Table 7. As can be seen, the model with infomax encoding outperforms the one without it in all cases. The performance gain becomes lower when the number of layers is higher. The reason is that when the number of layers is higher, (1) the number of sampled neighbors becomes lower, and (2) the smoothing effect of the GNN becomes stronger, leading to more homogeneity when infomax encoding is not used.

4.5 Additional analysis

In this section, we provide additional studies about (1) oversmoothing issue (2) training time, scalability, and convergence, (3) ablation study, (4) hyper-parameter sensitivity, and (5) adaptability of HIME to other frameworks.

4.5.1 Oversmoothing issue

First, we investigate whether HIME suffers from the oversmoothing issue, which is a common issue in GNNs. In addition, we aim to provide empirical evidences to support our motivation behind using GRUs. Therefore, we introduce two variants of the single-level aggregator by replacing GRU ($\mathbf{x}_i \leftarrow \text{GRU}(\mathbf{x}_i, \mathbf{n}_i)$) with GCN and RGCN styles: $\mathbf{x}_i \leftarrow \sigma(\omega_i \mathbf{W}\mathbf{x}_i + (1-\omega_i)\mathbf{n}_i)$, where σ denotes the rectified linear unit (ReLU), $\mathbf{W}$ is a learnable transformation matrix, and ω is a balancing factor between the information of



(a) DBLP **(b)** Yelp

Fig. 8 MAD values of HIME with different aggregation styles and layers. Lower MAD value indicates smoother the node representations in the graph. The GRU variant is the least prone to the oversmoothing issue

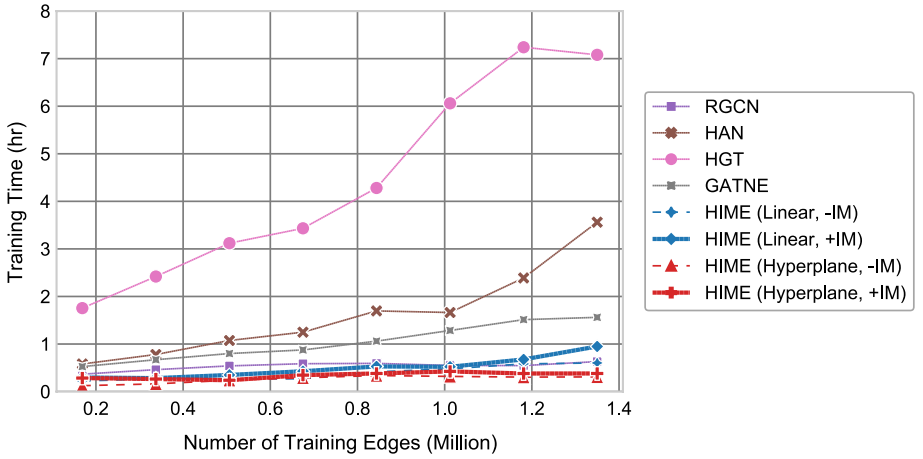


Fig. 9 Training time until convergence of each model with respect to the number of training edges. The number of layers is set to 1 for all models

a node and its neighbors that equals to $\frac{1}{|\mathcal{N}_i|+1}$ for GCN style and $\frac{1}{|\mathcal{R}|+1}$ for RGCN style. We conduct additional experiments for the GCN and RGCN styles by varying the number of layers L from 2 to 5 with the same setting as in Sect. 4.4.3.

The results are listed in Table 8. We observe that all variants achieve comparable results when $L = 2$. However, as the number of layers increases, the performance gap becomes larger on both datasets, and the GRU variant retains the performance the most among all of them. The results demonstrate the possibility of using GRU to alleviate the oversmoothing issue. To further investigate the difference among them, we use Mean Average Distance (MAD) between node representations (Chen et al., 2020) to measure the smoothness of the representations. The distance is defined as one minus the cosine similarity between the representations between a pair of nodes. Lower MAD value indicates smoother the node representations in a graph, where the zero value means that all the node representations become indistinguishable. The results are shown in Fig. 8. We observe significant drops in the MAD values of the GCN and RGCN variants on Yelp dataset indicating the possibility of oversmoothing. In contrast, the MAD values of the

Table 9 Performance (MRR) and time analysis in link prediction task on Amazon-Large dataset. "t/step" and "T" denotes time per step and total training time, respectively

	Macro	Micro	t/step (s)	T (h)
HIME (-IM)	86.73	78.54	.102	2.59
HIME	**87.32**	**79.34**	.109	2.71

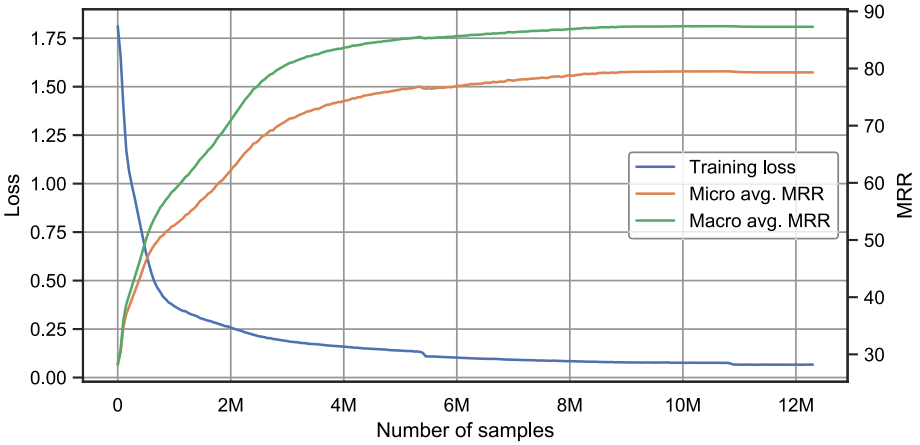


Fig. 10 Convergence curve for HIME on Amazon-Large dataset. The plotted performances are on the validation set

GRU slightly reduce when the number of layers increases on both datasets. This supports that HIME with GRU (default) does not suffer from the oversmoothing issue.

4.5.2 Training time, scalability, and convergence analysis

We aim to analyze the training time of HIME and compared it with bi-level aggregation models. We conduct experiments by training models on Douban Movie with varying the number of training edges. To see it clearly, we include the time for variants of HIME with varying the transformations h and with/without infomax encoding (IM). We plot the results in Fig. 9. As can be seen, the characteristics of the training time can be vary depending on models. For HIME, we find that its training time linearly grows with the number of edges, but the slope is smaller compared with GATNE, HAN, and HGT because of the less training steps until convergence. The faster convergence is probably due to the simplicity of single-level aggregations compared with those of bi-level aggregations employing attention mechanisms. On the other hand, RGCN demonstrates the similar training time as HIME because RGCN employs the mean pooling instead. .

Furthermore, to investigate the scalability of HIME, we run HIME on Amazon-Large dataset consisting of 12.8 million edges. We compare the link prediction performance of HIME to its variant without the infomax encoding. Table 9 shows the performance and its training time on a single GPU, NVIDIA Tesla V100. HIME with the infomax encoding performs better while costing a tiny amount of time. We plot the convergence curve

Table 10 Ablation study of HIME in comparison to RGCN in the link prediction task. Bold values indicate the best result for each dataset and metric

	DBLP		Yelp		Douban Movie		Douban Book	
	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
RGCN	65.29	66.16	74.06	70.56	60.20	62.85	58.74	57.85
HIME (i)	61.54	62.51	73.48	69.86	68.79	71.58	65.28	64.29
HIME (i,ii)	65.44	65.29	79.50	75.57	72.22	74.63	67.56	66.54
HIME (i,ii,iii)	**72.56**	**69.94**	**80.87**	**76.91**	**73.73**	**76.01**	**68.41**	**67.41**

of HIME in Fig. 10. We observe that the training loss is steadily converged and inversely proportional to the performance on the validation set.

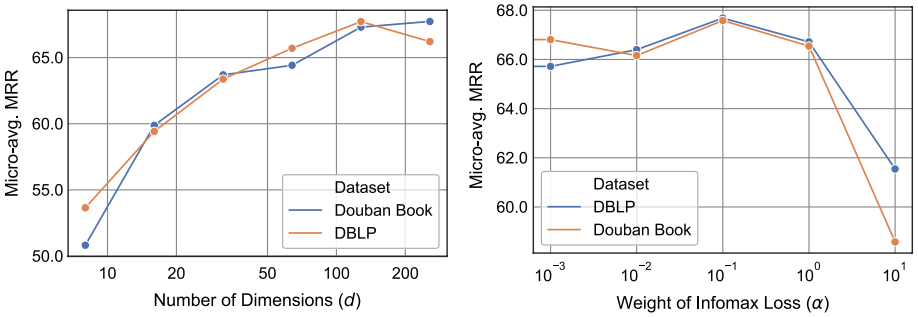
4.5.3 Ablation study

We provide ablation study to clarify the contributions of our work in the methodology over RGCN. Thus far, our contributions are as follows.

(i) We proposed single-level aggregation (Sect. 3.1), which uses mean pooling over all types of neighbors instead of averaging over each type of neighbors in RGCN.
(ii) We introduced the transformation h (Sect. 3.1) that should be carefully selected and fine-tuned, unlike RGCN that uses only linear transformation.
(iii) We proposed infomax encoding (Sect. 3.4), which can reduce the heterogeneity and promote the homogeneity between neighbors in the graph.

To demonstrate the effect of these three contributions, we perform ablation study of HIME by considering three variants of it that illustrate the contribution of each point on the top of its previous. Note that for (ii), we use the hyperplane transformation to distinguish from the linear transformation in RGCN.

The results are listed in Table 10. We observe that the effect of each contribution is significantly different and depends on the datasets. Specifically, the first contribution shows the highest performance gain over other contributions on Douban Movie and Douban Book datasets, where the average degree of a graph is high and the edges are heavily dominated by the majority relations (please see Table 12 in Appendix A for the statistics). Such a situation will cause a severe down-weighting issue in bi-level aggregations, and that explains the performance gain from the first contribution. Conversely, it performs significantly worse than RGCN on DBLP, where the average degree is much lower compared with Douban Movie and Douban Book datasets. This empirically supports the application of RGCN that was originally proposed for knowledge graphs, which usually have a small number of average degree. The second contribution positively yields significant improvements in all cases. Please be reminded that we have thoroughly investigated the effect of the third contribution in Sect. 4.4.



(a) Micro-average MRR w.r.t. d **(b)** Micro-average MRR w.r.t. α

Fig. 11 Parameter sensitivity

Table 11 Performance of HGT and HIME* on multiple tasks with and without pretrain. Bold values indicate the best result for each downstream task and metric

Task	HGT		HIME*	
	MRR	NDCG	MRR	NDCG
Without pretrain				
AD	57.96	79.82	**61.66**	**81.77**
PF	26.84	30.82	**34.94**	**34.97**
PV	16.39	32.88	**20.33**	**36.83**
With pretrain				
AD	**67.18**	**84.21**	65.88	84.08
PF	43.29	39.42	**45.17**	**41.16**
PV	24.61	41.90	**28.90**	**45.62**

4.5.4 Hyper-parameter sensitivity

We conduct parameter sensitivity analysis by adjusting important hyper-parameters: d and α with $L = 1$. We plot the model performance against them as shown in Fig. 11a, b. As d increases, the performance is raised until becoming plateau when $d > 100$. Then, the performance of the model is slightly affected by d when setting d high enough. On the other hand, the performance improves as α increases from 0.001 to 0.1, then it declines. HIME achieves the best results at around $\alpha = 0.1$, where the infomax loss positively contributes to the performance.

4.5.5 Adaptability of HIME to other frameworks

We investigate whether HIME can be used on other frameworks with a similar setting. We select GPT-GNN (Hu et al., 2020a) which is a framework for pretraining graph neural network and applicable for HINs. GPT-GNN aims to generate or reconstruct an input graph which is similar to the objective of HIME. The main difference between GPT-GNN and HIME is that GPT-GNN operates on subgraph sampling while HIME samples an observed edge, then generates relevant node representations in the training stage.

We conduct experiments based on GPT-GNN framework by comparing augmented HIME (HIME*) to HGT. HIME* uses relative temporal encoding, as in HGT, in relation-specific transformation to generate edge vectors. However, to keep the concept of single-level aggregation, HIME* does not use any attention. We use the same hyper-parameter setting[6] as being provided without tuning. We use Open Academic Graph (OAG) data on computer science (CS) provided by the GPT-GNN authors. we follow the original setting for time-transfer, then evaluate the models on three downstream tasks: prediction of Paper–Field (PF), Paper–Venue (PV), and Author Disambiguation (AD). Please see Sect. 4 in GPT-GNN literature for more details.

Table 11 shows the performance on the downstream tasks of both models. We observe that HIME* shows its superior over HGT with and without pretrain for most cases. This demonstrates that the concept of HIME, a heterogeneous single-level aggregator with infomax encoding, is applicable to other frameworks.

5 Conclusion

In this work, we proposed the single-level aggregation scheme and the application of infomax to learn node embedding for heterogeneous information networks. The single-level aggregation scheme is not only simpler than the bi-level scheme adopted by the state-of-the-art methods but it also has higher performance across many benchmark tests. The proposed infomax helps in bridging heterogeneous embedding to homogeneous embedding by encouraging graph smoothness and allows scalability. We conducted extensive experiments to verify and compare the performance of our model with the state-of-the-art methods.

Our results with single-level aggregation justify that the bi-level aggregation scheme down-weights some popular node types and edge types (such as users and user interactions) by design. In the light of this, it is beneficial to use our single-level aggregation scheme in future studies as a benchmark method.

For future direction, we aim to investigate a way to combine existing homogeneous GNNs and HIN embedding frameworks efficiently to accommodate a variety of homogeneous GNNs to HIN domain.

Appendix A: Additional statistics of datasets

Table 12 provides additional details including node types and relations. We choose to conduct experiments on ACM and IMDB with metapath pre-processing because of the extreme sparsity of the originals. For additional details about Amazon, YouTube, and Twitter datasets, please see the appendix of GATNE literature (Cen et al., 2019).

[6] Note that the corresponding authors of GPT-GNN and HGT are the same. We assume that this hyper-parameter setting is suitable for pretraining with HGT.

Table 12 Additional statistics of HIN datasets

Node type	Number of Nodes	Relation	Number of Edges
Toy			
User (U)	1,000	U-I	8,250
Item (I)	100	I-T	300
Tag (T)	10		
DBLP (Binbin Hu et al., 2019)			
Author (A)	14,475	A-P	41,794
Paper (P)	14,376	P-C	14,376
Conference (C)	20	P-T	114,624
Term (T)	8,920		
Yelp (Binbin Hu et al., 2019)			
User (U)	16,239	U-U	158,590
Business (B)	14,284	U-B	198,397
Category (C)	47	B-C	40,009
Location (L)	511	B-L	14,267
Douban Movie (Shi et al., 2019)			
User (U)	13,367	U-U	4,085
Group (G)	2,753	U-G	570,047
Movie (M)	12,677	U-M	1,068,278
Actor (A)	6,311	M-A	33,587
Director (D)	2,449	M-D	11,276
Douban Book (Shi et al., 2019)			
User (U)	13,024	U-U	169,150
Book (B)	22,347	U-B	1,584,124
Group (G)	2,936	U-G	2,378,542
Location (L)	6,311	U-L	21,184
Author (A)	453	B-A	22,836
Publisher (D)	1,815	B-P	43,546
Year (Y)	64	B-Y	42,384
Amazon-Large (Ni et al., 2019)			
User (U)	452,208	interact	7,557,648
Item (I)	598,935	also-view	2,395,093
		also-buy	2,844,386
ACM (Wang et al., 2019)			
Author	3,025	P-A-P	29,281
		P-S-P	2,210,761
IMDB (Wang et al., 2019)			
Movie	3,550	M-A-M	66,428
		M-D-M	13,788

Appendix B: Implementation notes and parameter settings

For a fair comparison, we fix the embedding dimensions d to 128. Without being mentioned, the number of negative samples is set to 5, and any other hyper-parameters are left with the default values as being provided. For DeepWalk, BHIN2vec, and metapath2vec,

we set the number of walks per node to 10, the walk length to 100 and the window size to 5. For knowledge graph embedding methods including **TransH** (Wang et al., 2014), **DistMult** (Yang et al., 2014), **ComplEx** (Trouillon et al., 2016), and **RotatE** (Sun et al., 2019) , we use the source code[7] provided by Sun et al., (2019). We use the binary cross-entropy loss for training DistMult and ComplEx. Note that ComplEx and RotatE embeds the node representations in a complex space $\mathbb{C}^d$, whereas other methods embeds them in a real space $\mathbb{R}^d$. For HAN and HGT, we set the number of multi-head attention to 8 and the output size of each head to 16. For a fair comparison between single-level and bi-level aggregations, we use the same hyper-parameter setting[8] for RGCN, HAN, HGT, and HIME without fine-tuning. For datasets which have no node attributes, all initial feature vectors $x_*^{(0)}$ are randomly initialized and learnable parameters.

1. **DeepWalk** (Perozzi et al., 2014): We perform random walk, then use Word2Vec in gensim[9] library, which is also used in the corresponding authors' code, to learn node representations.
2. **LINE** (Tang et al., 2015b): We set the number of samples to 10000M and initial learning rate to 0.01. We use the source code[10] provided by the authors to perform node embedding.
3. **BHIN2vec** (Lee et al., 2019b): We use the source code[11] provided by the authors for training the model. We set other hyper-parameters as the default setting. Specifically, we set $\alpha = 0.05$ and $r_2 = 0.0025$.
4. **metapath2vec** (Dong et al., 2017): To cover all node types and relations, we use meta-paths: ("APA", "APCPA", "PTP") for DBLP, ("UU", "UBU", "BCB", "BLB") for Yelp, ("UU", "UGU", "UMU", "MAM", "MDM") for Douban Movie and ("UBU", "UGU", "UU", "ULU", "BAB", "BPB", "BYB") for Douban Book. We performs meta-path-based random walk, then use Word2Vec to obtain node representations. We do not use the code from the authors because it is only for specific datasets, and is not applicable to others.
5. **HEER** (Shi et al., 2018b): We use the source code[12] from the authors for training the model. We take LINE-1st as pre-trained embedding for the model, followed the authors' setting.
6. **GraphSAGE** (Hamilton et al., 2017): We use the source code[13] provided by the authors. We choose mean aggregator to make it more closer to GATNE and HIME compared to other aggregators (max and LSTM). We set the number of layers to 2, and the numbers of sampled neighbors are set to 50 and 20 for one- and two-hop neighbors, respectively. It is optimized via Adam with learning rate 0.0001.
7. **DGI** (Veličković et al., 2018): We use the source code provided by DMGI authors since it is more suitable for multiplex networks than that provided by the corresponding authors.

[7] https://github.com/DeepGraphLearning/KnowledgeGraphEmbedding

[8] We do not apply the same hyper-parameter setting to GATNE because it uses distinct node features in aggregation and optimizes another objective: heterogeneous skip-gram.

[9] https://radimrehurek.com/gensim/

[10] https://github.com/tangjianpku/LINE

[11] https://github.com/sh0416/BHIN2VEC

[12] https://github.com/GentleZhu/HEER

[13] https://github.com/williamleif/GraphSAGE

8. **DMGI** (Park et al., 2020): We use the source code[14] from the corresponding authors. Since the original liturature uses the same datsets: ACM and IMDB, We use the same setting as provided: dropout, *consensus regularization*, and l_2 regularization to 0.5, 0.001, and 0.0001, respectively.
9. **GATNE** (Cen et al., 2019): We use the source code[15] from the corresponsing authors. Since its performance is surprisingly low for link prediction in HINs, we try tuning many hyper-parameters: heterogeneous skip-gram, edge embedding dimensions, and early stopping criteria. None of these yields positively noticeable change. We report the results with its default hyper-parameter setting: numbers of edge embedding dimensions, attention dimensions, sampled neighbors, and negative samples are 10, 20, 10, and 5, respectively. The same hyper-parameter setting is applied to GATNE*.
10. **RGCN** (Schlichtkrull et al., 2018): We re-implement the model based on the code provided by corresponding authors and the implementation in PyG library due to performance and scalablity issues.
11. **HAN** (Wang et al., 2019): We re-implement the model based on the code provided by corresponding authors since we use RGCN-based for unsupervised learning. Generally, the aggregation of HAN can be understood as multiple GAT convolution modules for distinct metapaths (or relations) with *semantic* attention for combining the outputs in the second-level. We use relations instead of metapaths which require supervision. For semi-supervised HAN [†], use the implementation[16] in Deep Graph Library[17] (DGL).
12. **HGT** (Hu et al., 2020b): Since the code provided by the corresponding authors uses different settings[18] than us, we use RGCN-based for unsupervised training instead. For a fair comparison to other models, we do not use layer normalization. Since all datasets have no time information, we turn off *relative temporal encoding*.
13. **HIME**: We intuitively set number of layers and sampled neighbors N to 1 and 50, respectively. From that, we find the suitable α from $\{10, 1, 0.1, 0.01, 0.001\}$ and heterogeneous projection h from {*hyperplane translation*[19] (Wang et al., 2014), *linear transformation*} via grid search. We found that $\alpha = 0.1$ performs best in all cases. For node clustering and classification tasks, we set dropout to 0.6.

Running environment

The experiments are conducted on Intel(R) Xeon(R) Gold 6148 CPU @ 2.4 GHz, 20 Cores, 60GB RAM, NVIDIA Tesla V100-16GB. We implement our model with PyTorch[20] 1.5 in Python 3.6. Alternatively, we have implemented our model in PyTorch Geometric library[21] (Fey and Lenssen, 2019) for future benchmarking.

[14] https://github.com/pcy1302/DMGI
[15] https://github.com/THUDM/GATNE
[16] https://github.com/dmlc/dgl/tree/master/examples/pytorch/han
[17] https://github.com/dmlc/dgl
[18] In their implementation, the model is optimized on each downstream task instead of learning node representation in general.
[19] Mathematically, $h(x_j, r) = \mathbf{x}_j - \mathbf{w}_r^{\intercal}\mathbf{x}_j\mathbf{w}_r$,, where $\mathbf{w}_r \in \mathbb{R}^d$, $\|\mathbf{w}_r\| = 1$, is a relation-specific hyperplane.
[20] https://pytorch.org
[21] https://github.com/rusty1s/pytorch_geometric

Author Contributions NC: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing—original draft, Writing—review and editing, Visualization. XL: Resources, Writing—original draft, Writing—review and editing, Supervision, Funding acquisition. NTH: Writing—original draft, Writing—Review and Editing, Visualization. TM: Resources, Writing—review and editing, Supervision, Funding acquisition.

Funding This work was partly supported by JSPS Grant-in-Aid for Scientific Research (Grant Number 21K12042, 17H01785), JST CREST (Grant Number JPMJCR1687), and the New Energy and Industrial Technology Development Organization (Grant Number JPNP20006). Nuttapong Chairatanakul was supported by MEXT scholarship.

Availability of data and material The data that support the findings of this study are publicly available and can be obtained from Binbin Hu et al., (2019), Shi et al., (2019), Ni et al., (2019), Wang et al., (2019), Cen et al., (2019), and Hu et al., (2020a).

Code availability The source code will be publicly released upon successful submission.

Declarations

Conflict of interest The authors declare that they have no known cofinancial interests or personal relationships that could have appeared to influence the work reported in this paper.

Ethics approval Not Applicable.

Consent to participate Not Applicable.

Consent for publication Not Applicable.

References

Bianchi, F., Tagliabue, J., Yu, B., Bigon, L., & Greco, C. (2020). Fantastic embeddings and how to align them: Zero-shot inference in a multi-shop scenario. In: *Proceedings of the SIGIR 2020 eCom Workshop*. arXiv:2007.14906

Binbin Hu, Y.F., & Shi, C. (2019). Adversarial learning on heterogeneous information network. In: *Proceedings of the 25th ACM SIGKDD conference on knowledge discovery and data mining, ACM*.

Bordes, A., Usunier, N.., Garcia-Durán, A., Weston, J., & Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. In: *Proceedings of the 26th International Conference on Neural Information Processing Systems—Volume 2*, Curran Associates Inc., Red Hook, NY, USA, NIPS'13, pp. 2787–2795.

Cao, S., Lu, W., & Xu, Q. (2015). GraRep: Learning graph representations with global structural information. In: *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management—CIKM '15*, ACM Press, Melbourne, Australia, pp. 891–900.

Cen, Y., Zou, X., Zhang, J., Yang, H., Zhou, J., & Tang, J. (2019). Representation learning for attributed multiplex heterogeneous network. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining—KDD '19*, ACM Press, Anchorage, AK, USA, pp 1358–1368.

Chairatanakul, N., Murata, T., & Liu, X. (2019). Recurrent translation-based network for top-n sparse sequential recommendation. *IEEE Access, 7*, 131567–131576. https://doi.org/10.1109/ACCESS.2019.2941083

Chen, D., Lin, Y., Li, W., Li, P., Zhou, J., & Sun, X. (2020). Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In: *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, AAAI Press, New York, NY, USA, pp. 3438–3445.

Chen, H., Yin, H., Wang, W., Wang, H., Nguyen, Q.V.H., & Li, X. (2018). PME: Projected Metric Embedding on Heterogeneous Networks for Link Prediction. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ACM Press, London, United Kingdom, pp. 1177–1186.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, Doha, Qatar, pp. 1724–1734.

Do, K., Tran, T., Nguyen, T., & Venkatesh, S. (2019). Attentional multilabel learning over graphs: A message passing approach. *Machine Learning, 108*(10), 1757–1781.

Dong, Y., Chawla, N. V., & Swami, A. (2017). metapath2vec: Scalable representation learning for heterogeneous networks. In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining—KDD '17*, ACM Press, Halifax, NS, Canada, pp. 135–144.

Feng, F., He, X., Tang, J., & Chua, T. S. (2019). *Graph adversarial training: Dynamically regularizing based on graph structure*. IEEE: IEEE Transactions on Knowledge and Data Engineering Publisher.

Fey, M., & Lenssen, J. E. (2019). Fast graph representation learning with PyTorch Geometric. In: *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

Ty, Fu., Lee, W. C., & Lei, Z. (2017). HIN2Vec: Explore meta-paths in heterogeneous information networks for representation learning. In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management—CIKM '17*, ACM Press, Singapore, Singapore, pp. 1797–1806

Goyal, P., & Ferrara, E. (2018). Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems, 151*, 78–94.

Grbovic, M., Radosavljevic, V., Djuric, N., Bhamidipati, N., Savla, J., Bhagwan, V., & Sharp, D. (2015). E-commerce in your inbox: Product recommendations at scale. In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Association for Computing Machinery, New York, NY, USA, KDD '15, pp. 1809–1818

Grover, A., & Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining—KDD '16*, ACM Press, San Francisco, California, USA, pp. 855–864

Hamilton, W. L., Ying, R., & Leskovec, J. (2017). Inductive representation learning on large graphs. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Curran Associates Inc., Red Hook, NY, USA, NIPS'17, pp. 1025–1035

Hjelm, R. D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., & Bengio, Y. (2018). Learning deep representations by mutual information estimation and maximization. In: ICLR'19.

Hu, B., Fang, Y., & Shi, C. (2019). Adversarial learning on heterogeneous information networks. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM Press, New York, NY, USA, KDD '19, pp. 120–129

Hu, Z., Dong, Y., Wang, K., Chang, K. W., & Sun, Y. (2020a). GPT-GNN: Generative pre-training of graph neural networks. In: *Proceedings of the 26th ACM SIGKDD conference on knowledge discovery and data mining, ACM*

Hu, Z., Dong, Y., Wang, K., & Sun, Y. (2020b). Heterogeneous graph transformer. In: *Proceedings of The Web Conference 2020*, ACM Press, New York, NY, USA, WWW '20, pp. 2704–2710

Inokuchi, A., Washio, T., & Motoda, H. (2003). Complete mining of frequent patterns from graphs: Mining graph data. *Machine Learning, 50*(3), 321–354.

Jin, H., & Zhang, X. (2019). Latent adversarial training of graph convolution networks. In: *ICML Workshop on Learning and Reasoning with Graph-Structured Representations*

Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. In: *International Conference on Learning Representations*

Kong, X., Yu, P. S., Ding, Y., & Wild, D. J. (2012). Meta path-based collective classification in heterogeneous information networks. In: *Proceedings of the 21st ACM international conference on Information and knowledge management—CIKM '12*, ACM Press, Maui, Hawaii, USA, p 1567.

Lazaridou, A., Pham, N. T., & Baroni, M. (2015). Combining language and vision with a multimodal skip-gram model. In: *Proceedings of the 2015 Conference of the North American Chapter of the*

Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, Denver, Colorado, pp. 153–163

Lee, N., Ajanthan, T., & Torr, P.H. (2019a) Snip: Single-shot network pruning based on connection sensitivity. In: ICLR.

Lee, S., Park, C., & Yu, H. (2019b). BHIN2vec: Balancing the type of relation in heterogeneous information network. In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, ACM Press, Beijing, China, CIKM '19, pp. 619–628

Li, Q., Han, Z., & Wu, X.m. (2018). Deeper insights into graph convolutional networks for semi-supervised learning. In: AAAI-18, vol 32.

Liu, X., Yu, Y., Guo, C., & Sun, Y. (2014). Liu X, Yu Y, Guo C, Sun Y (2014) Meta-Path-Based Ranking with Pseudo Relevance Feedback on Heterogeneous Graph for Citation Recommendation. In: *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management—CIKM '14*, ACM Press, Shanghai, China, pp. 121–130

Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In: *Advances in neural information processing systems*, pp. 3111–3119.

Miller, H. J., & Han, J. (2001). *Geographic data mining and knowledge discovery*. USA: Taylor & Francis Inc.

Ng, A.Y., Jordan, M.I., & Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. In: *Advances in neural information processing systems*, pp. 849–856.

Ni, J., Li, J., & McAuley, J. (2019). Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In: *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, Association for Computational Linguistics, Hong Kong, China, pp. 188–197.

NT, H., & Maehara, T. (2019). Revisiting graph neural networks: All we have is low-pass filters. arXiv preprint arXiv:190509550

Oono, K., & Suzuki, T. (2019). Graph neural networks exponentially lose expressive power for node classification. In: *ICLR*.

Park, C., Kim, D., Han, J., & Yu, H. (2020). Unsupervised attributed multiplex network embedding. In: *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, AAAI Press, New York, NY, USA, pp. 5371–5378

Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). DeepWalk: online learning of social representations. In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining—KDD '14*, ACM Press, New York, NY, USA, pp. 701–710.

Qiu, J., Dong, Y., Ma, H., Li, J., Wang, K., & Tang, J. (2018). Network embedding as matrix factorization: unifying DeepWalk, LINE, PTE, and node2vec. In: *Proceedings of the 11th ACM International Conference on Web Search and Data Mining—WSDM '18*, ACM Press, Los Angeles, California, USA, pp. 459–467

Ren, Y., Liu, B., Huang, C., Dai, P., Bo, L., & Zhang, J. (2019). Heterogeneous deep graph infomax. arXiv preprint arXiv:191108538

Rendle, S., Freudenthaler, C., Gantner, Z., & Schmidt-Thieme, L. (2009). BPR: Bayesian personalized ranking from implicit feedback. In: Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence, AUAI Press, Arlington, Virginia, USA, UAI '09, pp 452–461

Ricci, F., Rokach, L., Shapira, B., & Kantor, P. B. (2010). *Recommender Systems Handbook* (1st ed.). New York Inc, New York: Springer-Verlag.

Schlichtkrull, M., Kipf, T. N., Bloem, P., Van Den Berg, R., Titov, I., & Welling, M. (2018). Modeling relational data with graph convolutional networks. In: *European Semantic Web Conference*, Springer, pp. 593–607.

Shi, C., Hu, B., Zhao, W., & Yu, P. S. (2019). Heterogeneous information network embedding for recommendation. *IEEE Transactions on Knowledge and Data Engineering, 31*(02), 357–370.

Shi, Y., Gui, H., Zhu, Q., Kaplan, L., & Han, J. (2018a). Aspem: Embedding learning by aspects in heterogeneous information networks. In: *Proceedings of SIAM International Conference on Data Mining (SDM18)*, San Diego, California, USA, pp. 144–152

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

Shrikumar, A., Greenside, P., & Kundaje, A. (2017). Learning important features through propagating activation differences. In: *Proceedings of the 34th International Conference on Machine Learning—Volume 70*, ICML'17, pp. 3145–3153.

Sun, Y., & Han, J. (2013). Mining heterogeneous information networks: A structural analysis approach. *SIGKDD Explorations Newsletter, 14*(2), 20–28.

Sun, Y., Han, J., Yan, X., Yu, P. S., & Wu, T. (2011). Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *Proceedings of the VLDB Endowment, 4*(11), 992–1003.

Sun, Y., Norick, B., Han, J., Yan, X., Yu, P. S., & Yu, X. (2012). Integrating meta-path selection with user-guided object clustering in heterogeneous information networks. In: Explorations Newsletter Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining—KDD '1212, ACM Press, Beijing, China, p. 1348.

Sun, Z., Deng, Z. H., Nie, J. Y., & Tang, J. (2019). RotatE: Knowledge graph embedding by relational rotation in complex space. In: *International Conference on Learning Representations*.

Tang, J., Qu, M., & Mei, Q. (2015a). PTE: Predictive text embedding through large-scale heterogeneous text networks. In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM Press, Sydney, NSW, Australia, KDD '15, pp. 1165–1174.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., & Mei, Q. (2015b). LINE: Large-scale information network embedding. In: *Proceedings of the 24th International Conference on World Wide Web—WWW '15*, ACM Press, Florence, Italy, pp. 1067–1077.

Trouillon, T., Welbl, J., Riedel, S., Gaussier, E., & Bouchard, G. (2016). Complex embeddings for simple link prediction. *International Conference on Machine Learning (ICML), 48*, 2071–2080.

Tsoumakas, G., & Katakis, I. (2007). Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM), 3*(3), 1–13.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). Graph attention networks. In: *International Conference on Learning Representations.*

Veličković, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., & Hjelm, R. D. (2018). Deep graph infomax. In: *ICLR'19*.

Wang, D., Cui, P., & Zhu, W. (2016). Structural deep network embedding. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining—KDD'16*, ACM Press, San Francisco, California, USA, pp. 1225–1234.

Wang, X., Ji, H., Shi, C., Wang, B., Ye, Y., Cui, P., & Yu, P. S. (2019). Heterogeneous graph attention network. In: *Proceedings of The Web Conference 2019*, ACM Press, San Francisco, CA, USA, pp. 2022–2032.

Wang, Z., Zhang, J., Feng, J., & Chen, Z. (2014). Knowledge graph embedding by translating on hyperplanes. In: *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, AAAI Press, Québec City, Québec, Canada, AAAI'14, pp. 1112–1119.

Wu, W., Li, B., Chen, L., & Zhang, C. (2018). Efficient attributed network embedding via recursive randomized hashing. In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, AAAI Press, IJCAI'18, pp. 2861–2867.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P.S. (2019). A comprehensive survey on graph neural networks. arXiv preprint arXiv:190100596

Xie, Y., Li, S., Yang, C., Wong, R. C. W., & Han, J. (2020). When do GNNs work: Understanding and improving neighborhood aggregation. In: Bessiere, C.(ed.) *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pp. 1303–1309.

Xu, L., Wei, X., Cao, J., & Yu, P.S. (2017). Embedding of Embedding (EOE): Joint Embedding for Coupled Heterogeneous Networks. In: *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining—WSDM '17, ACM Press*, Cambridge, United Kingdom, pp. 741–749.

Yang, B., Wt, Yih, He, X., Gao, J., & Deng, L. (2014). Embedding entities and relations for learning and inference in knowledge bases. In: *International Conference on Learning Representations*.

Yang, D., Rosso, P., Li, B., & Cudre-Mauroux, P. (2019). Nodesketch: Highly-efficient graph embeddings via recursive sketching. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Association for Computing Machinery*, New York, NY, USA, KDD '19, pp. 1162–1172, https://doi.org/10.1145/3292500.3330951

Yang, Z., Ding, M., Zhou, C., Yang, H., Zhou, J., & Tang, J. (2020). Understanding negative sampling in graph representation learning. *In: KDD,'20*, pp. 1666–1676.

Yun, S., Jeong, M., Kim, R., Kang, J., & Kim, H. J. (2019). Graph transformer networks. *Advances in Neural Information Processing Systems, 32* (pp. 11983–11993). Vancouver, Canada: Curran Associates Inc.

Zhu Y, Xu Y, Cui H, Yang C, Liu Q, Wu S (2021) Structure-aware hard negative mining for heterogeneous graph contrastive learning. arXiv:210813886 [cs]