Chapter 3
Structure-preserved Heterogeneous Graph Representation

Abstract Heterogeneous graph (HG) contains various types of nodes or links, which are highly-correlated and present intricate structures due to different links. These structures reflect the crucial factors of topology. Therefore encoding meaningful structures is a basic requirement to obtain node representations with high-quality. So far, some representative structures have been studied in an HG, from one-hop edges to high-order local structures, such as meta-paths and network schema. In this chapter, we will introduce several works focusing on structure preservation. By capturing respective structures, they successfully depict the rich semantics and complex heterogeneity, and effectively support downstream tasks.

3.1 Introduction

One basic requirement of graph representation learning is to preserve the graph structure properly. A lot of early works focus on the structural preservation of homogeneous graph representation learning, which focus on how to maintain the second-order [43], higher-order [3], and community structure [48] in homogeneous graphs, and therefore have achieved good results in downstream tasks such as node classification and link prediction. Compared with homogeneous graphs, there are more challenges in maintaining structural information for heterogeneous graphs (HGs), because the latter contain multiple types of nodes and multiple relations among nodes.

Traditional heterogeneous graph modeling often uses meta-paths [38] to measure the structural similarity of nodes. However, these meta-path based methods cannot calculate the similarity of nodes without meta-path connections, which greatly limits the application scenarios of these methods. Inspired by homogeneous graph representation learning methods, many heterogeneous graph representation learning methods have recently been proposed, and the latent heterogeneous network embeddings have been further applied to various network mining tasks, such as node classication [18], clustering [40, 41], and similarity search [39, 59]. In contrast to

conventional meta-path based methods [38], the advantage of latent-space representation learning lies in its ability to model structural similarities between nodes without connected meta-paths.

In this chapter, we will introduce several representative works which preserve structural information in node representations. In Section 3.2 and 3.3, we will first introduce a **H**eterogeneous graph **E**mbedding based **Rec**ommendation model (named **HERec**) [35] and a **Neu**ral network based **A**spect-level **C**ollaborative **F**iltering model (named **NeuACF**) [11], which combine meta-path with two classic homogeneous graph representation learning methods separately, that is, skip-gram algorithm and attention-based deep neural network. In Section 3.4, we will introduce a **R**elation-structure aware **H**eterogeneous **I**nformation **N**etwork **E**mbedding model (named **RHINE**) [23], which divides the relations in heterogeneous graphs into two categories based on mathematical analysis, thereby modeling the structural information in the heterogeneous graphs in a more fine-grained manner. In Section 3.5, We will introduce a novel **N**etwork **S**chema preserving **H**eterogeneous graph **E**mbedding method (named **NSHE**) [62], which uses the network schema, that is, the meta template of the heterogeneous graph, to model the structure information more completely, and get rid of the dependence on the hand-designed meta-paths.

3.2 Meta-path based Random Walk

3.2.1 Overview

In a graph, there exist many kinds of structures to depict the different relations between nodes, where neighborhood structure, describing the local structure of a node, is an important one. How to appropriately reserve the neighborhood structure in node representations is a fundamental problem. To solve the problem, a typical technique usually used is random walk, which is a process that starts from a target node and randomly chooses a neighbor of the last node as the current one [10, 28]. By exploiting this mechanism, multiple random sequences for nodes are generated, and each describes a local structure over original graph. We can view each sequence as a sentence, and nodes in the sequence as words. In this way, word2vec-based framework proposed in natural language processing (NLP) is adopted to learn node representations, which captures the proximity between neighbor nodes in a window with fixed size.

However, these works focus on representation learning for homogeneous graph, only considering singular type of nodes and relationships. In the field of HG, it is a challenge to effectively combine random walk strategy with complex heterogeneity to capture the local semantic structure. To tackle the challenge, **H**G **E**mbedding based **Rec**ommendation (named **HERec**) [35] designs a novel mechanism to per-

form random walk along meta-paths, also called meta-path based random walk, considering that meta-paths are sufficient to depict rich semantics.

Different from random walk in homogeneous graph, HERec constrains generated node sequence aligned with predefined metapath. For example, for a meta-path Author-Paper-Author (APA), if type of the last selected node i is A, then HERec chooses a neighbor of i with type P. The objective of HERec is to simultaneously learn representations of multiple types of nodes and preserve both the structures semantics of a given HG. Due to its effectiveness, HERec, combined with classic matrix factorization framework, is also applied in recommender systems. A basic recommender system usually consists users and items, and aims to help users discover latent items of interest. By the designed framework, HERec effectively extracts semantic and local information from interactions between users and items, and gives a good performance in downstream tasks.

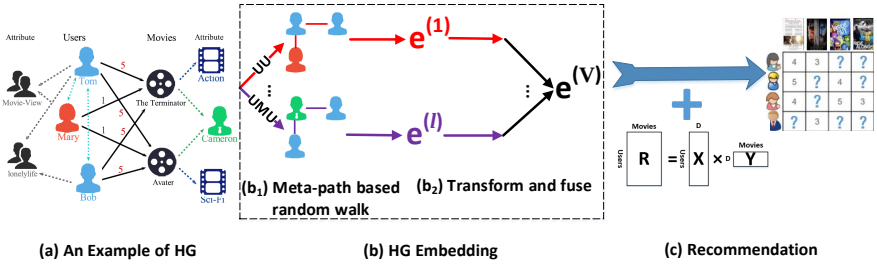


Fig. 3.1 The schematic illustration of the proposed HERec approach.

3.2.2 The HERec Model

3.2.2.1 Model Framework

In this section, we provide more details about HERec here. Specifically, HERec is designed for the HG in a recommender system, which consists of two major components. The first part is to learn user/item embeddings from HG, namely HG embedding (Fig. 3.1b). The second part is to involve the learnt embeddings into classic matrix factorization framework, namely recommendation (Fig. 3.1c). Next, we will present detailed illustration of HERec.

3.2.2.2 Heterogeneous Graph Embedding

Given an HG $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, the goal is to learn a low-dimensional embedding $\boldsymbol{e}_v \in \mathbb{R}^d$ for each node $v \in \mathcal{V}$, which is much easier to be used and integrated in subsequent procedures, compared with meta-path based similarity. Inspired by deepwalk [28], a well-known embedding method for homogeneous graph, HERec designs a novel random walk mechanism over HG to learn embeddings.

Meta-path based Random Walk To capture the complex heterogeneity, meta-path is usually adopted to characterize the semantic patterns for HGs [38]. Therefore, HERec also employs a meta-path based random walk method to generate node sequences. Giving a heterogeneous graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ and a meta-path $\rho : A_1 \xrightarrow{R_1} \cdots A_t \xrightarrow{R_t} A_{t+1} \cdots \xrightarrow{R_l} A_{l+1}$, the walk path is generated according to the following distribution:

$$
\begin{aligned}
& P(n_{t+1} = x | n_t = v, \rho) \\
= & \begin{cases} \frac{1}{|\mathcal{N}^{A_{t+1}}(v)|}, & (v,x) \in \mathcal{E} \text{ and } \phi(x) = A_{t+1}; \\ 0, & \text{otherwise}, \end{cases}
\end{aligned} \tag{3.1}
$$

where n_t is the t-th node in the walk, the type of v is A_t, $\phi(\cdot)$ is the object type mapping function, and $\mathcal{N}^{(A_{t+1})}(v)$ is the first-order neighbor set for node v with the type of A_{t+1}. A walk will follow the pattern of a meta-path repetitively until it reaches the pre-defined length.

Example 1 *We still take Fig. 3.1a as an example, which represents a movie recommender system. Given a meta-path UMU, we can generate two sample walks (i.e. node sequences) by starting with the user* Tom*: (1)* $Tom_{User} \rightarrow The\ Terminator_{Movie} \rightarrow Mary_{User}$*, and (2)* $Tom_{User} \rightarrow Avater_{Movie} \rightarrow Bob_{User} \rightarrow The\ Terminator_{Movie} \rightarrow Mary_{User}$*. Similarly, given the meta-path UMDMU, we can also generate another node sequence:* $Tom_{User} \rightarrow The\ Terminator_{Movie} \rightarrow Cameron_{Director} \rightarrow Avater_{Movie} \rightarrow Mary_{User}$*. It is intuitive to see that these meta-paths can lead to meaningful node sequences corresponding to different semantic relations.*

Type Constraint and Filtering Consider that in recommendation, only user and item should be more emphasized than other types of nodes. So, only meta-paths starting with *user type* or *item type* are selected. To further strengthen user and item, other types of nodes that are different from the starting type will be filtered out from the generated node sequences. In this way, the final sequences only contain one type of nodes (i.e. user of item). Here are two advantages. First, there is only one type, which relaxes the challenging goal of representing all the heterogeneous objects in a unified space. Second, given a fixed-length window, a node is able to utilize more homogeneous neighbors that are more likely to be relevant than others with different types.

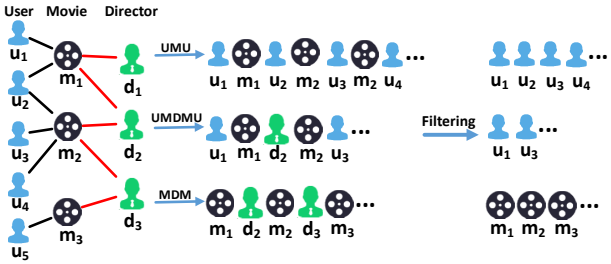


Fig. 3.2 An illustrative example of the proposed meta-path based random walk. First perform random walks guided by some selected meta-paths, and then filter out nodes of types that are different from the starting type.

Example 2 *As shown in Fig. 3.2, only the meta-paths starting from* user type *or* item type *are derived, such as UMU, UMDMU and MUM. Take the meta-path of UMU as an instance. We can generate a sampled sequence "*$u_1 \rightarrow m_1 \rightarrow u_2 \rightarrow m_2 \rightarrow u_3 \rightarrow m_2 \rightarrow u_4$*" according to Eq. 3.1. And then, we remove the movies and finally obtain a homogeneous node sequence "*$u_1 \rightarrow u_2 \rightarrow u_3 \rightarrow u_4$*".*

After this step, the next focus will be how to learn effective embeddings for nodes in sequences.

Optimization Objective Given a processed node sequence, HERec defines the neighborhood for node u based on co-occurrence in a fixed-length window in the sequence, denotes as $\mathcal{N}_u$. Following node2vec [10], the final embeddings of nodes are learnt by optimizing the following objective:

$$\max_{f} \sum_{u \in \mathcal{V}} \log Pr(\mathcal{N}_u | f(u)), \tag{3.2}$$

where $f : \mathcal{V} \rightarrow \mathbb{R}^d$ is a function mapping each node to d-dimensional embedding space. $Pr(\mathcal{N}_u | f(u))$ measures probability of u's neighbors given u's embedding. The embedding mapping function $f(\cdot)$ is trained by utilizing stochastic gradient descent (SGD). A major difference between previous method [7] and HERec lies in the construction of $\mathcal{N}_u$, where HERec selects homogeneous neighbors using meta-path based random walks.

Embedding Fusion For one meta-path l and corresponding node sequence, after optimizing via Eq. 3.2, given a node $v \in \mathcal{V}$, we can obtain its embedding $\boldsymbol{e}_v^{(l)}$ based on meta-path l. In a similar way, we can obtain a set of embeddings $\{\boldsymbol{e}_v^{(l)}\}_{l=1}^{|\mathcal{P}|}$, where $\mathcal{P}$ denotes the set of meta-paths. Then, we propose to use a general function $g(\cdot)$, which aims to fuse the learnt node embeddings to get $\boldsymbol{e}_u^{(U)}$ and $\boldsymbol{e}_i^{(I)}$ for users and items, respectively:

$$\begin{aligned} \boldsymbol{e}_u^{(U)} &\leftarrow g(\{\boldsymbol{e}_u^{(l)}\}), \\ \boldsymbol{e}_i^{(I)} &\leftarrow g(\{\boldsymbol{e}_i^{(l)}\}). \end{aligned} \tag{3.3}$$

There are three fusion functions that can be used to fuse these embeddings. We discuss the cases for users below. The functions for items are similar and can be derived accordingly.

- *Simple linear fusion*. We simply average the results from different meta-paths after transforming the embeddings with MLP.

$$g(\{\boldsymbol{e}_u^{(l)}\}) = \frac{1}{|\mathcal{P}|}\sum_{l=1}^{|\mathcal{P}|}(\mathbf{M}^{(l)}\boldsymbol{e}_u^{(l)} + \boldsymbol{b}^{(l)}), \tag{3.4}$$

  where $\mathbf{M}^{(l)} \in \mathbb{R}^{D\times d}$ and $\boldsymbol{b}^{(l)} \in \mathbb{R}^D$ are the transformation matrix and bias vector *w.r.t.* the l-th meta-path.
- *Personalized linear fusion*. We further assign each user with a weight vector on the meta-paths, representing the user's personalized preference for each meta-path.

$$g(\{\boldsymbol{e}_u^{(l)}\}) = \sum_{l=1}^{|\mathcal{P}|} w_u^{(l)}(\mathbf{M}^{(l)}\boldsymbol{e}_u^{(l)} + \boldsymbol{b}^{(l)}), \tag{3.5}$$

  where $w_u^{(l)}$ is the learnt preference weight of user u over the l-th meta-path.
- *Personalized non-linear fusion*. Moreover, we use non-linear function to enhance the fusion ability.

$$g(\{\boldsymbol{e}_u^{(l)}\}) = \sigma\Big(\sum_{l=1}^{|\mathcal{P}|} w_u^{(l)}\sigma\big(\mathbf{M}^{(l)}\boldsymbol{e}_u^{(l)} + \boldsymbol{b}^{(l)}\big)\Big), \tag{3.6}$$

  where $\sigma(\cdot)$ is a non-linear function, i.e., sigmoid function or others.

Among the three fusion strategies, HERec demonstrates that personalization and non-linearity are essential for boosting the results. The parameters involved in $g(\cdot)$ are optimized together with the recommendation model, which is introduced in the next section.

3.2.2.3 Integrating Matrix Factorization with Fused HG Embedding for Recommendation

Rating predictor is a component used to predict the ratings of the users on the items. In MF, the rating $r_{u,i}$ of a user u on an item i is simply defined as follows:

$$\widehat{r_{u,i}} = \mathbf{x}_u^\top \cdot \mathbf{y}_i, \tag{3.7}$$

where $\mathbf{x}_u \in \mathbb{R}^D$ and $\mathbf{y}_i \in \mathbb{R}^D$ denote the latent factors corresponding to user u and item i by factorizing the user-item rating matrix. Since we have also obtained the embeddings for user u and item i, we can further incorporate them into the rating predictor as below:

$$\widehat{r_{u,i}} = \mathbf{x}_u^\top \cdot \mathbf{y}_i + \alpha \cdot \boldsymbol{e}_u^{(U)^\top} \cdot \gamma_i^{(I)} + \beta \cdot \gamma_u^{(U)^\top} \cdot \boldsymbol{e}_i^{(I)}, \tag{3.8}$$

where $\gamma_u^{(U)}$ and $\gamma_i^{(I)}$ are user-specific and item-specific latent factors to pair with the HG embeddings $\boldsymbol{e}_i^{(I)}$ and $\boldsymbol{e}_u^{(U)}$ respectively, and α and β are the tuning parameters to integrate the three terms.

The overall objective is formulated as follows:

$$\begin{aligned} \pounds = & \sum_{\langle u,i,r_{u,i}\rangle \in \mathcal{R}} (r_{u,i} - \widehat{r_{u,i}})^2 + \lambda \sum_u (\|\mathbf{x}_u\|_2 + \|\mathbf{y}_i\|_2 \\ & + \|\boldsymbol{\gamma}_u^{(U)}\|_2 + \|\boldsymbol{\gamma}_i^{(I)}\|_2 + \|\boldsymbol{\Theta}^{(U)}\|_2 + \|\boldsymbol{\Theta}^{(I)}\|_2), \end{aligned} \tag{3.9}$$

where $\widehat{r_{u,i}}$ is the predicted rating using Eq. 3.8 by HERec, λ is the regularization parameter, and $\boldsymbol{\Theta}^{(U)}$ and $\boldsymbol{\Theta}^{(I)}$ are the parameters of the function $g(\cdot)$ for users and items respectively. SGD is utilized to efficiently train HERec.

3.2.3 Experiments

3.2.3.1 Experimental Settings

Datasets Three real datasets are used to perform downstream tasks, consisting of Douban Movie dataset [1] from the movie domain, Douban Book dataset [2] from the book domain, and Yelp dataset [3] from the business domain. The detailed descriptions of the three datasets are shown in Table 3.1. It should be pointed out that these three datasets also have different rating sparsity degrees: the Yelp dataset is very sparse, while the Douban Movie dataset is much denser.

Baselines To demonstrate the effectiveness, HERec is compared with three categories of baselines: classic MF based rating prediction methods { *PMF* [27], *SoMF* [24] }, and HG based recommendation methods { FM_{HIN} [30], *HeteMF* [58], *SemRec* [37] and *DSR* [63] }. Moreover, there are two variants of HERec: HERec_{dw} and HERec_{mp}, where the difference is the way of random walk during generating node sequences. The former adopts deepwalk [28] to obtain node embeddings,

[1] http://movie.douban.com

[2] http://book.douban.com

[3] http://www.yelp.com/dataset-challenge

Table 3.1 Statistics of the three datasets.

Dataset (Density)	Relations (A-B)	Number of A	Number of B	Number of (A-B)	Ave. degrees of A	Ave. degrees of B	Meta-paths
Douban Movie (0.63%)	User-Movie	13,367	12,677	1,068,278	79.9	84.3	UMU, MUM UMDMU, MDM UMAMU, MAM UMTMU, MTM
	User-User	2,440	2,294	4,085	1.7	1.8	
	User-Group	13,337	2,753	570,047	42.7	207.1	
	Movie-Director	10,179	2,449	11,276	1.1	4.6	
	Movie-Actor	11,718	6,311	33,587	2.9	5.3	
	Movie-Type	12,678	38	27,668	2.2	728.1	
Douban Book (0.27%)	User-Book	13,024	22,347	792,026	60.8	35.4	UBU, BUB UBPBU, BPB UBYBU, BYB UBABU
	User-User	12,748	12,748	169,150	13.3	13.3	
	Book-Author	21,907	10,805	21,905	1.0	2.0	
	Book-Publisher	21,773	1,815	21,773	1.0	11.9	
	Book-Year	21,192	64	21,192	1.0	331.1	
Yelp (0.08%)	User-Business	16,239	14,284	198,397	12.2	13.9	UBU, BUB UBCiBU, BCiB UBCaBU, BCaB
	User-User	10,580	10,580	158,590	15.0	15.0	
	User-Compliment	14,411	11	76,875	5.3	6988.6	
	Business-City	14,267	47	14,267	1.0	303.6	
	Business-Category	14,180	511	40,009	2.8	78.3	

which ignores the heterogeneity of nodes. The later employs metapath2vec++ [7] to generate node embeddings, which does not filter out different types of nodes.

3.2.3.2 Effectiveness Experiments

In this experiment, we set different training ratios for each dataset, and use MAE and RMSE as metrics to evaluate the performance. We set the embedding dimension as 64 and randomly run experiments 10 times and report the average results shown in Table 3.2. Notice that, HERec adopts the personalized non-linear fusion function here.

From the results, we can draw the following conclusions: (1) HERec is consistently better than other baselines because it provides better information extraction (a new HGs embedding model) and utilization (an extended MF model). (2) HG based methods, especially FM_{HIN}, perform better than traditional MF based methods, which indicates the usefulness of the heterogeneous information. (3) On one hand, compared with HERec$_{mp}$, HERec$_{dw}$ performs much worse, which indicates HG embedding methods are important to HG based recommendation again. On the other hand, HERec outperforms HERec$_{mp}$, which demonstrates that it is more effective to perform task-specific HG embedding for improving the recommendation performance (e.g. only focus on user and item).

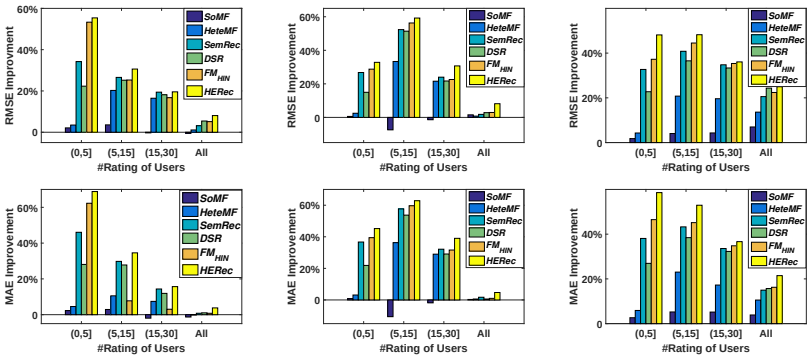
3.2.3.3 Cold-start Prediction

Now, we check the performances of all the methods in cold-start prediction settings, where there are fewer rating records but heterogeneous context information is available. First according to the counts of users' rating records, they are devided into

Table 3.2 Results of effectiveness experiments on three datasets. A smaller MAE or RMSE value indicates a better performance. For easier interpretation, the improvement of each method w.r.t. the PMF model is also reported. A larger improvement ratio indicates a better performance.

Dataset	Training	Metrics	PMF	SoMF	FM_{HIN}	HeteMF	SemRec	DSR	$HERec_{dw}$	$HERec_{mp}$	HERec
Douban Movie	80%	MAE	0.5741	0.5817	0.5696	0.5750	0.5695	0.5681	0.5703	**0.5515**	0.5519
		Improve		-1.32%	+0.78%	-0.16%	+0.80%	+1.04%	+0.66%	+3.93%	+3.86%
		RMSE	0.7641	0.7680	0.7248	0.7556	0.7399	0.7225	0.7446	0.7121	**0.7053**
		Improve		-0.07%	+5.55%	+1.53%	+3.58%	+5.85%	+2.97%	+7.20%	+8.09%
	60%	MAE	0.5867	0.5991	0.5769	0.5894	0.5738	0.5831	0.5838	0.5611	**0.5587**
		Improve		-2.11%	+1.67%	-0.46%	+2.19%	+0.61%	+0.49%	+4.36%	+4.77%
		RMSE	0.7891	0.7950	0.7842	0.7785	0.7551	0.7408	0.7670	0.7264	**0.7148**
		Improve		-0.75%	+0.62%	+1.34%	+4.30%	+6.12%	+2.80%	+7.94%	+9.41%
	40%	MAE	0.6078	0.6328	0.5871	0.6165	0.5945	0.6170	0.6073	0.5747	**0.5699**
		Improve		-4.11%	+3.40%	-1.43%	+2.18%	-1.51%	+0.08%	+5.44%	+6.23%
		RMSE	0.8321	0.8479	0.7563	0.8221	0.7836	0.7850	0.8057	0.7429	**0.7315**
		Improve		-1.89%	+9.10%	+1.20%	+5.82%	+5.66%	+3.17%	+10.71%	+12.09%
	20%	MAE	0.7247	0.6979	0.6080	0.6896	0.6392	0.6584	0.6699	0.6063	**0.5900**
		Improve		+3.69%	+16.10%	+4.84%	+11.79%	+9.14%	+7.56%	+16.33%	+18.59%
		RMSE	0.9440	0.9852	0.7878	0.9357	0.8599	0.8345	0.9076	0.7877	**0.7660**
		Improve		-4.36%	+16.55%	+0.88%	+8.91%	+11.60%	+3.86%	+16.56%	+18.86%
Douban Book	80%	MAE	0.5774	0.5756	0.5716	0.5740	0.5675	0.5740	0.5875	0.5591	**0.5502**
		Improve		+0.31%	+1.00%	+0.59%	+1.71%	+0.59%	-1.75%	+3.17%	+4.71%
		RMSE	0.7414	0.7302	0.7199	0.7360	0.7283	0.7206	0.7450	0.7081	**0.6811**
		Improve		+1.55%	+2.94%	+0.77%	+1.81%	+2.84%	-0.44%	+4.53%	+8.17%
	60%	MAE	0.6065	0.5903	0.5812	0.5823	0.5833	0.6020	0.6203	0.5666	**0.5600**
		Improve		+2.67%	+4.17%	+3.99%	+3.83%	+0.74%	-2.28%	+6.58%	+7.67%
		RMSE	0.7908	0.7518	0.7319	0.7466	0.7505	0.7552	0.7905	0.7318	**0.7123**
		Improve		+4.93%	+7.45%	+5.59%	+5.10%	+4.50%	+0.04%	+7.46%	+9.93%
	40%	MAE	0.6800	0.6161	0.6028	0.5982	0.6025	0.6271	0.6976	0.5954	**0.5774**
		Improve		+9.40%	+11.35%	+12.03%	+11.40%	+7.78%	-2.59%	+12.44%	+15.09%
		RMSE	0.9203	0.7936	0.7617	0.7779	0.7751	0.7730	0.9022	0.7703	**0.7400**
		Improve		+13.77%	+17.23%	+15.47%	+15.78%	+16.01%	+1.97%	+16.30%	+19.59%
	20%	MAE	1.0344	0.6327	0.6396	0.6311	0.6481	0.6300	1.0166	0.6785	**0.6450**
		Improve		+38.83%	+38.17%	+38.99%	+37.35%	+39.10%	+1.72%	+34.41%	+37.65%
		RMSE	1.4414	0.8236	0.8188	0.8304	0.8350	0.8200	1.3205	0.8869	**0.8581**
		Improve		+42.86%	+43.19%	+42.39%	+42.07%	+43.11%	+8.39%	+38.47%	+40.47%
Yelp	90%	MAE	1.0412	1.0095	0.9013	0.9487	0.9043	0.9054	1.0388	0.8822	**0.8395**
		Improve		+3.04%	+13.44%	+8.88%	+13.15%	+13.04%	+0.23%	+15.27%	+19.37%
		RMSE	1.4268	1.3392	1.1417	1.2549	1.1637	1.1186	1.3581	1.1309	**1.0907**
		Improve		+6.14%	+19.98%	+12.05%	+18.44%	+21.60%	+4.81%	+20.74%	+23.56%
	80%	MAE	1.0791	1.0373	0.9038	0.9654	0.9176	0.9098	1.0750	0.8953	**0.8475**
		Improve		+3.87%	+16.25%	+10.54%	+14.97%	+15.69%	+0.38%	+17.03%	+21.46%
		RMSE	1.4816	1.3782	1.1497	1.2799	1.1771	1.1208	1.4075	1.1516	**1.1117**
		Improve		+6.98%	+22.40%	+13.61%	+20.55%	+24.35%	+5.00%	+22.27%	+24.97%
	70%	MAE	1.1170	1.0694	0.9108	0.9975	0.9407	0.9429	1.1196	0.9043	**0.8580**
		Improve		+4.26%	+18.46%	+10.70%	+15.78%	+15.59%	-0.23%	+19.04%	+23.19%
		RMSE	1.5387	1.4201	1.1651	1.3229	1.2108	1.1582	1.4632	1.1639	**1.1256**
		Improve		+7.71%	+24.28%	+14.02%	+21.31%	+24.73%	+4.91%	+24.36%	+26.85%
	60%	MAE	1.1778	1.1135	0.9435	1.0368	0.9637	1.0043	1.1691	0.9257	**0.8759**
		Improve		+5.46%	+19.89%	+11.97%	+18.18%	+14.73%	+0.74%	+21.40%	+25.63%
		RMSE	1.6167	1.4748	1.2039	1.3713	1.2380	1.2257	1.5182	1.1887	**1.1488**
		Improve		+8.78%	+25.53%	+15.18%	+23.42%	+24.19%	+6.09%	+26.47%	+28.94%

three groups, i.e. $(0,5]$, $(5,15]$ and $(15,30]$. It is easy to see that the case for the first group is the most difficult, since users from this group have fewest rating records. Here, HERec is only compared with HG based recommendation, including SoMF, HeteMF, SemRec, DSR and FM_{HIN}, and the results are shown in Fig. 3.3, which reports the improvement ratios w.r.t. PMF. Overall, all the comparison methods are better than PMF (i.e., show positive y-axis values). Moreover, HERec performs the



(a) Douban Movie (b) Douban Book (c) Yelp

Fig. 3.3 Performance comparison of different methods for cold-start prediction on three datasets. *y*-axis denotes the improvement ratio over PMF.

best among all the methods, and the improvement over PMF becomes more significant for users with fewer rating records. The results indicate that HG based information is effective to improve the recommendation performance, and HERec can effectively utilize HG information in a more principled way.

The more detailed method description and experiment validation can be seen in [35].

3.3 Meta-path based Decomposition

3.3.1 Overview

In the last section, an HG embedding method exploiting meta-path based random walk is introduced, which generates node sequences that follow the given meta-paths for optimizing the similarity between nodes. However, an observation is that the properties of an object in an HG may stem from different aspects due to the rich type information, which poses a challenge for heterogeneous graph representation. How to effectively extract and fuse different semantic aspect-level information plays an important role in HG representation, which isn't considered thoroughly in previous random walk-based methods [35, 7, 9].

To this end, meta-path based decomposition based techniques aim to decompose HG into several sub-graphs according to different meta-paths, each representing a specific semantic aspect, and preserve the proximity of nodes in each sub-graph. **NeuACF** (**Neu**ral network based **A**spect-level **C**ollaborative **F**iltering) [11, 34] is a piece of representative work which exploits decomposition-based HG representation

to learn the aspect-level representations and effectively fuse them for recommendation.

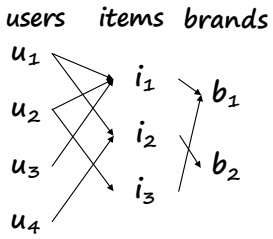


Fig. 3.4 A toy example of an HG with different aspect-level information.

Fig. 3.4 illustrates an HG composed of 3 types of nodes, which are *User*, *Item* and *Brand*; and 2 types of relations, which are *User-Item* purchase relation and *Item-Brand* indicating which brand a given item belongs to. Given such an HG, we target to learn the representations of the users and items, and make item recommendation for the users. In this HG, there exist purchase-aspect and brand-aspect information which should be both effectively captured. We can learn the representations of user nodes from the aspect of purchase history with the *User-Item-User* path. Meanwhile, we can also learn the representations from the aspect of brand preference with the *User-Item-Brand-Item-User* path. Furthermore, a delicate deep network is designed to learn different aspect-level representations and an attention mechanism is utilized to effectively fuse them for top-N recommendation. Comparing to the above method NeuACF, we further propose NeuACF++ to fuse aspect information with self-attention mechanism which considers different aspect-level representations and learns the attention values simultaneously. More details about NeuACF and NeuACF++ are given in the following subsections.

3.3.2 The NeuACF Model

3.3.2.1 Model Framework

The basic idea of NeuACF is to extract different aspect-level representations for users and items, and then learn and fuse these representations with deep neural network. The model contains three major steps, the architecture of which is illustrated in Fig. 3.5. First, an HG is constructed based on the rich user-item interaction information, and the aspect-level similarity matrices are computed under different meta-paths of HG which reflect different aspect-level features of users and items. Next, a deep neural network is designed to learn the aspect-level representations separately by taking these similarity matrices as inputs. Finally, the aspect-level representations

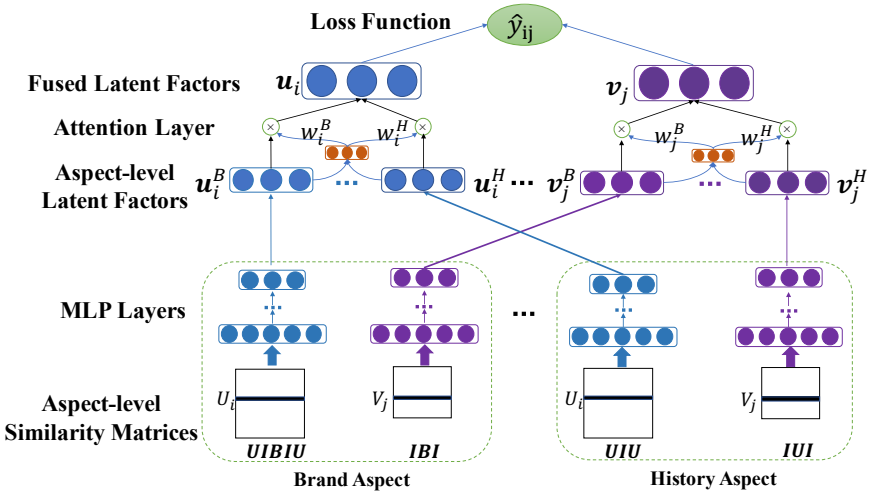


Fig. 3.5 The architecture of the NeuACF model.

are combined with an attention component to obtain the overall representations for users and items. Moreover, we also employ self-attention mechanism to fuse aspect-level representations more effectively, extending the model as NeuACF++, which is introduced in Section 3.3.2.5. Next we elaborate on the three steps.

3.3.2.2 Aspect-level Similarity Matrix Extraction

We employ HG to organize objects and relations, due to its power of information fusion and semantics representation [37]. Furthermore, we utilize meta-path to decompose the HG and extract different aspect-level features of users and items.

Given a specific meta-path, similarity matrix is employed to extract the aspect-level features. The popular PathSim [38] is employed to calculate aspect-level similarity matrices under different meta-paths in experiments. For example, the similarity matrices of user-user and item-item are computed based on the meta-path $UIBIU$ and IBI for the brand-aspect features.

The computation of similarity matrix based on meta path is of great importance in the proposed model, so how to compute similarity matrix quickly is an important problem in the method. In real-world applications, the complexity of similarity matrix computation is not high because the similarity matrix is usually very sparse for most meta paths. Based on this fact, there are several acceleration computation methods proposed by previous works [36, 38] for similarity matrix computation, for example, PathSim-pruning [38], dynamic programming strategy and Monte Carlo (MC) strategy [36]. Moreover, there are also many new methods for similarity matrix computation, for example, BLPMC [52], PRSim [50]. In addition, the similarity matrix can be computed offline in advance in our model. The similarity matrix

is computed with training data, so we can prepare the similarity matrix before the training processing.

3.3.2.3 Learning Aspect-level Representations

With the computed user-user and item-item similarity matrices of different aspects, their representations are next learned. A deep neural network is designed to learn their corresponding aspect-level representations separately. Concretely, as shown in Fig. 3.5, for each user in each aspect, the user's similarity vector is extracted from the aspect-specific similarity matrix. Then we take the similarity vector as the input of the MLP and MLP learns the aspect-level representation as the output. The item representations of each aspect can be learned similarly.

Taking the similarity matrix $\boldsymbol{S}^B \in \mathbb{R}^{N \times N}$ of users under the meta-path $UIBIU$ as an example, user U_i is represented as an N-dimensional vector $\boldsymbol{S}^B_{i*}$, which means the similarities between U_i and all the other users. Here N means the total number of users in the dataset.

The MLP projects the initial similarity vector $\boldsymbol{S}^B_{i*}$ of user U_i to a low-dimensional aspect-level representation. In each layer of MLP, the input vector is mapped into another vector in a new space. Formally, given the initial input vector $\boldsymbol{S}^B_{i*}$, and the l-th hidden layer $\boldsymbol{H}_l$, the final aspect-level representation $\boldsymbol{u}^B_i$ can be learned through the following multi-layer mapping functions.

From the learning framework in Fig. 3.5, one can see that for each aspect-level similarity matrix of both users and items there is a corresponding MLP learning component described above to learn the aspect-level representations. Since there are various meta-paths connecting users and items, different aspect-level representations can be learned.

3.3.2.4 Attention based Aspect-level Representations Fusion

After the aspect-level representations are learned separately for users and items, next we need to integrate them together to obtain aggregated representations. A straightforward way is to concatenate all the aspect-level representations to form a higher-dimensional vector. Another intuitive way is to average all the representations. The issue is that both methods do not distinguish their different importance because not all the aspects contribute to the model performance equally, which will be showed in the experiment part.

Therefore, the attention mechanism is chosen to fuse these aspect-level representations. Attention mechanism has shown the effectiveness in various machine learning tasks such as image captioning and machine translation [53, 57, 1]. The advantage of attention mechanism is that it can learn to assign attentive values (normalized by sum to 1) for all the aspect-level representations: higher (lower) values indicate that the corresponding features are more informative (less informative) for recommendation. Specifically, given the user's brand-aspect representations $\boldsymbol{u}^B_i$, a

two-layer network is used to compute the attention score $\boldsymbol{s}_i^B$ by Eq. 3.10,

$$\boldsymbol{s}_i^B = \boldsymbol{W}_2^T f\left(\boldsymbol{W}_1^T \cdot \boldsymbol{u}_i^B + \boldsymbol{b_1}\right) + \boldsymbol{b_2}, \tag{3.10}$$

where $\boldsymbol{W}_*$ and $\boldsymbol{b}_*$ are the weight matrices and the biases, respectively, and we use the $ReLU$, i.e., $f(x) = max(0,x)$ as the activation function.

The final attention values for the aspect-level representations are obtained by normalizing the above attentive scores with the Softmax function given in Eq. 3.11, which can be interpreted as the contributions of different aspects B to the aggregated latent factor of user U_i,

$$\boldsymbol{w}_i^B = \frac{exp(\boldsymbol{s}_i^B)}{\sum_{A=1}^{L} exp(\boldsymbol{s}_i^A)}, \tag{3.11}$$

where L is the total number of all the aspects.

After obtaining all the attention weights $\boldsymbol{w}_i^B$ of all the aspect-level representations for user U_i, the aggregated representation $\boldsymbol{u}_i$ can be calculated by Eq. 3.12,

$$\boldsymbol{u}_i = \sum_{B=1}^{L} \boldsymbol{w}_i^B \cdot \boldsymbol{u}_i^B. \tag{3.12}$$

This attention method is adopted by NeuACF in the experiments.

3.3.2.5 NeuACF++: Self-Attention based Aspect-level Representations Fusion

Recently, self-attention mechanism has received considerable research interests. For example, Vaswani et al. [46] and Devlin et al. [6] utilize self-attention to learn the relationship between two sequences. Learning dependencies and relationships between aspect-level representations is the most important part in our model, and self-attention has ability to model the relationships between the different aspect-level representations. Therefore, we extend the standard attention mechanism of NeuACF to self-attention, and call the extension version of the model NeuACF++. Next, the self-attention mechanism employed in NeuACF++ will be introduced.

Different from standard attention mechanism, self-attention mainly focuses on the co-learning attentions of two sequences. The vanilla attention mechanism mainly considers computing the attention values based on the user or item representations of one aspect, while self-attention mechanism is able to learn the attention values from different aspects simultaneously. For example, the Brand-level representation of users have strong relationship to the Brand-level representation of items, and the self-attention mechanism can learn this relationship and promote the model performance. So the learned values are able to capture more information on the multi-aspects.

Specifically, we firstly compute the affinity scores between all aspect-level representations. For a user U_i, the affinity score of two different aspect-level representations $\boldsymbol{u}_i^B$ and $\boldsymbol{u}_i^C$ can be calculated by their inner product:

$$M_i^{B,C} = (\boldsymbol{u}_i^B)^T \cdot \boldsymbol{u}_i^C. \tag{3.13}$$

The matrix $\boldsymbol{M}_i = [M_i^{B,C}] \in \mathbb{R}^{L \times L}$ is also called the self-attention matrix, where L is the total number of aspects. There is an affinity matrix $\boldsymbol{M}_i$ for each user. Basically, the matrix $\boldsymbol{M_i}$ characterizes the similarity of aspect-level representations for a specific user U_i, which reflects the correlation between two aspects when recommending for this user. When the aspect B is equal to aspect C, $M_i^{B,C}$ will get a high value due to the inner product operator, so a zero mask is added to avoid a high matching score between identical vectors.

The aspect-level representations learned from self-attention mechanism are not independent. Users can make a trade-off between those aspects. The affinity matrix measures the importance of different aspect-level representations, so the representation of aspect B for the specific user i is computed based on the self-attention matrix as:

$$\boldsymbol{g}_i^B = \sum_{C=1}^{L} \frac{exp(\boldsymbol{M}_i^{B,C})}{\sum_{A=1}^{L} exp(\boldsymbol{M}_i^{B,A})} \boldsymbol{u}_i^C. \tag{3.14}$$

Then for all the aspects, we can obtain the final representation of users or items as:

$$\boldsymbol{u}_i = \sum_{B=1}^{L} \boldsymbol{g}_i^B. \tag{3.15}$$

The self-attention mechanism can learn self-attentive representations from different aspect-level information effectively. In order to distinguish with the above attention method NeuACF, the self-attention mechanism is implemented as NeuACF++ in the experiments.

3.3.2.6 Objective Function

We model the top-N recommendation as a classification problem which predicts the probability of interaction between users and items in the future. In order to ensure that the output value is a probability, we need to constrain the predicted score $\hat{y}_{ij}$ in the range of [0,1], where we use a Logistic function as the activation function for the output layer. The probability of the interaction between the user U_i and item I_j is calculated according to Eq. 3.16,

$$\hat{y}_{ij} = sigmod(\boldsymbol{u}_i \cdot \boldsymbol{v}_j) = \frac{1}{1 + e^{-\boldsymbol{u}_i \cdot \boldsymbol{v}_j}}, \tag{3.16}$$

where $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$ are the aggregated representations of user U_i and item I_j respectively.

Over all the training set, according to the above settings, the likelihood function is:

$$p(\mathcal{Y}, \mathcal{Y}^- | \Theta) = \prod_{i,j \in \mathcal{Y}} \hat{y}_{ij} \prod_{i,k \in \mathcal{Y}^-} (1 - \hat{y}_{ik}), \tag{3.17}$$

where $\mathcal{Y}$ and $\mathcal{Y}^-$ are the positive and negative instances sets, respectively. The negative instance set $\mathcal{Y}^-$ is sampled from unobserved data for training. Θ is the parameters set.

Since the ground truth y_{ij} is in the set $\{0,1\}$, Eq. 3.17 can be rewritten as:

$$p(\mathcal{Y},\mathcal{Y}^-|\Theta) = \prod_{i,j\in\mathcal{Y}\cup\mathcal{Y}^-} (\hat{y}_{ij})^{y_{ij}}(1-\hat{y}_{ij})^{(1-y_{ij})}. \tag{3.18}$$

Then we can take the negative logarithm of the likelihood function to get the point-wise loss function in Eq. 3.19:

$$Loss = -\sum_{i,j\in\mathcal{Y}\cup\mathcal{Y}^-} (y_{ij} log \hat{y}_{ij} + (1-y_{ij}) log(1-\hat{y}_{ij})), \tag{3.19}$$

This is the overall objective function of the model, and it can be optimized by stochastic gradient descent or its variants [19].

3.3.3 Experiments

3.3.3.1 Experimental Settings

Datasets The proposed model is evaluated over the publicly available MovieLens dataset [12] and Amazon dataset [13, 25]. We use the origin Movielens dataset for the experiment. For Amazon dataset, we remove the users who bought less than 10 items.

Evaluation Metrics We adopt the leave-one-out method [14] for evaluation. The latest rated item of each user is held out for testing, and the remaining data for training. Following previous work [14], we randomly select 99 items that are not rated by the users as negative samples and rank the 100 sampled items for the users. For a fair comparison with the baseline methods, we use the same negative sample set for each (*user, item*) pair in the test set for all the methods. We evaluate the model performance through the Hit Ratio (HR) and the Normalized Discounted Cumulative Gain (NDCG) defined in Eq. 3.20,

$$HR = \frac{\#hits}{\#users}, NDCG = \frac{1}{\#users}\sum_{i=1}^{\#users}\frac{1}{log_2(p_i+1)}, \tag{3.20}$$

where $\#hits$ is the number of users whose test item appears in the recommended list and p_i is the position of the test item in the list for the i-th hit. In the experiments, we truncate the ranked list at $K \in [5,10,15,20]$ for both metrics.

Baselines Besides two basic methods (i.e., ItemPop and ItemKNN [32]), the baselines include two MF methods (MF [21] and eALS [15]), one pairwise rank-

ing method (BPR [31]), and two neural network based methods (DMF [55] and NeuMF [14]). In addition, we use SVD_{hin} to leverage the heterogeneous information for recomendation, and also adopt two recent HG based methods (FMG [61] and HeteRS [29]) as baselines.

3.3.3.2 Performance Analysis

Table 3.3 HR@K and NDCG@K comparisons of different methods.

Datasets	Metrics	ItemPop	ItemKNN	MF	eALS	BPR	DMF	NeuMF	SVD_{hin}	HeteRS	FMG	NeuACF	NeuACF++
ML100K	HR@5	0.2831	0.4072	0.4634	0.4698	0.4984	0.3483	0.4942	0.4655	0.3747	0.4602	0.5097	**0.5111**
	NDCG@5	0.1892	0.2667	0.3021	0.3201	0.3315	0.2287	0.3357	0.3012	0.2831	0.3014	0.3505	**0.3519**
	HR@10	0.3998	0.5891	0.6437	0.6638	0.6914	0.4994	0.6766	0.6554	0.5337	0.6373	0.6846	**0.6915**
	NDCG@10	0.2264	0.3283	0.3605	0.3819	0.3933	0.2769	0.3945	0.3988	0.3338	0.3588	0.4068	**0.4092**
	HR@15	0.5366	0.7094	0.7338	0.7529	0.7741	0.5873	0.7635	0.7432	0.6524	0.7338	0.7813	**0.7832**
	NDCG@15	0.2624	0.3576	0.3843	0.4056	0.4149	0.3002	0.4175	0.4043	0.3652	0.3844	0.4318	**0.4324**
	HR@20	0.6225	0.7656	0.8144	0.8155	0.8388	0.6519	0.8324	0.8043	0.7224	0.8006	**0.8464**	0.8441
	NDCG@20	0.2826	0.3708	0.4034	0.4204	0.4302	0.3151	0.4338	0.3944	0.3818	0.4002	**0.4469**	**0.4469**
ML1M	HR@5	0.3088	0.4437	0.5111	0.5353	0.5414	0.4892	0.5485	0.4765	0.3997	0.4732	**0.5630**	0.5584
	NDCG@5	0.2033	0.3012	0.3463	0.3670	0.3756	0.3314	0.3865	0.3098	0.2895	0.3183	**0.3944**	0.3923
	HR@10	0.4553	0.6171	0.6896	0.7055	0.7161	0.6652	0.7177	0.6456	0.5758	0.6528	0.7202	**0.7222**
	NDCG@10	0.2505	0.3572	0.4040	0.4220	0.4321	0.3877	0.4415	0.3665	0.3461	0.3767	0.4453	**0.4454**
	HR@15	0.5568	0.7118	0.7783	0.7914	0.7988	0.7649	0.7982	0.7689	0.6846	0.7536	0.8018	**0.8030**
	NDCG@15	0.2773	0.3822	0.4275	0.4448	0.4541	0.4143	0.4628	0.4003	0.3749	0.4034	**0.4667**	0.4658
	HR@20	0.6409	0.7773	0.8425	0.8409	0.8545	0.8305	0.8586	0.8234	0.7682	0.8169	0.8540	**0.8601**
	NDCG@20	0.2971	0.3977	0.4427	0.4565	0.4673	0.4296	0.4771	0.4456	0.3947	0.4184	0.4789	**0.4790**
Amazon	HR@5	0.2412	0.1897	0.3027	0.3063	0.3296	0.2693	0.3117	0.3055	0.2766	0.3216	0.3268	**0.3429**
	NDCG@5	0.1642	0.1279	0.2068	0.2049	0.2254	0.1848	0.2141	0.1922	0.1800	0.2168	0.2232	**0.2308**
	HR@10	0.3576	0.3126	0.4278	0.4287	0.4657	0.3715	0.4309	0.4123	0.4207	0.4539	0.4686	**0.4933**
	NDCG@10	0.2016	0.1672	0.2471	0.2441	0.2693	0.2179	0.2524	0.2346	0.2267	0.2595	0.2683	**0.2792**
	HR@15	0.4408	0.3901	0.5054	0.5065	0.5467	0.4328	0.5258	0.5056	0.5136	0.5430	0.5591	**0.5948**
	NDCG@15	0.2236	0.1877	0.2676	0.2647	0.2908	0.2332	0.2774	0.2768	0.2513	0.2831	0.2924	**0.3060**
	HR@20	0.4997	0.4431	0.5680	0.5702	0.6141	0.4850	0.5897	0.5607	0.5852	0.6076	0.6257	**0.6702**
	NDCG@20	0.2375	0.2002	0.2824	0.2797	0.3067	0.2458	0.2925	0.2876	0.2683	0.2983	0.3080	**0.3236**

The proposed methods are applied to recommendation task. Table 3.3 shows the experiment results of different methods. The proposed methods are marked as NeuACF which implements the attention method in Section 3.3.2.4 and NeuACF++ which implements the self-attention mechanism in Section 3.3.2.5, respectively. One can draw the following conclusions.

Firstly, NeuACF and NeuACF++ achieve all the best performance over all the datasets and criteria. The improvement of the two models comparing to these baselines is significant. This indicates that the aspect level information is useful for the downstream task. Besides, NeuACF++ outperforms the NeuACF method in most circumstances. Particularly, the performance of NeuACF++ is significantly improved in Amazon dataset (about +2% at HR and +1% at NDCG). This demonstrates the effectiveness of the self-attention mechanism. Since the affinity matrix evaluates the similarity score of different aspects, we can extract the valuable information from the aspect representations.

Secondly, NeuMF, as one neural network based method, also performs well on most conditions, while both NeuACF and NeuACF++ outperform NeuMF in almost all the cases. The reason is probably that multiple aspects of representations learned

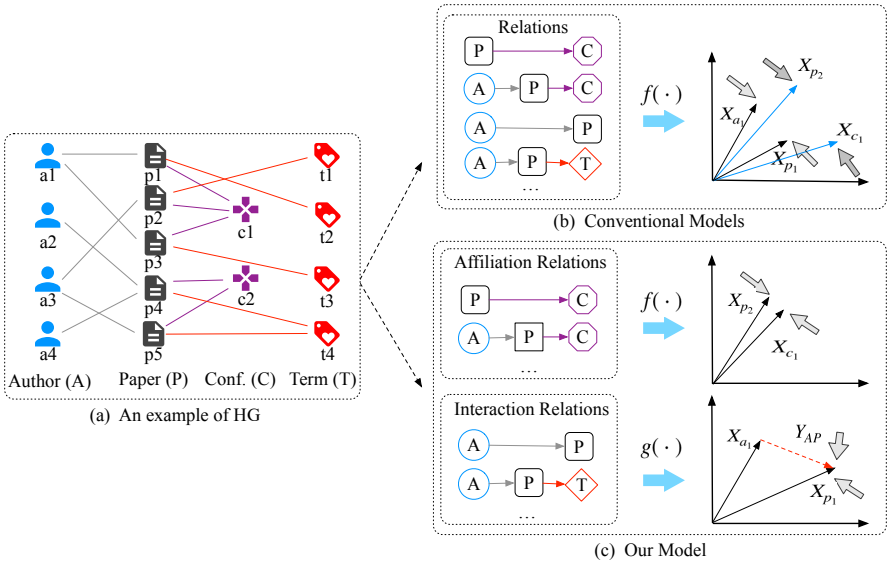


Fig. 3.6 The illustration of HGs and the comparison between conventional methods and our method (non-differentiated relations v.s. differentiated relations).

by NeuACF and NeuACF++ provide more features of users and items. Although FMG also utilizes the same features as NeuACF and NeuACF++, the better performance of NeuACF and NeuACF++ implies that the deep neural network and the attention mechanisms in NeuACF and NeuACF++ may have the better ability to learn representations of users and items than the "shallow" model in FMG.

The more detailed method description and experiment validation can be seen in [11] and [34].

3.4 Relation Structure Awareness

3.4.1 Overview

To model the heterogeneity of networks, several attempts have been done on HGs embedding. Representative methods include random walk based methods [33, 7, 9], decomposion based methods [42, 54, 35], and neural network based methods [4, 60, 47, 11]. Although these methods consider the heterogeneity of networks, they usually have an assumption that one single model can handle all relations and nodes, through keeping the representations of two nodes close to each other, as illustrated in Fig. 3.6b.

However, various relations in HGs have significantly different structural characteristics, which should be handled with different models. Let us see a toy example

in Fig. 3.6a. The relations in the network include atomic relations (e.g., AP and PC) and composite relations (e.g., APA and APC). Intuitively, AP relation and PC relation reveal rather different structures. That is, some authors write some papers in the AP relation, which shows a peer-to-peer structure. While that many papers are published in one conference in the PC relation reveals the structure of one-centered-by-another. Similarly, APA and APC indicate peer-to-peer and one-centered-by-another structures respectively. The intuitive examples clearly illustrate that relations in HGs indeed have different structural characteristics.

In this section, we present a novel model for HGs embedding, named **R**elation structure-aware **H**eterogeneous **I**nformation **N**etwork **E**mbedding (**RHINE**) [23]. In specific, we first explore the structural characteristics of relations in HGs with thorough mathematical analysis, and present two structure-related measures which can consistently distinguish the various relations into two categories: Affiliation Relations (ARs) with one-centered-by-another structures and Interaction Relations (IRs) with peer-to-peer structures, as shown in Fig. 3.6c. In order to capture the distinctive structural characteristics of the relations, we then propose two specifically designed models. For ARs where the nodes share similar properties [56], we calculate Euclidean distance as the proximity between nodes, so as to make the nodes directly close in the low-dimensional space. On the other hand, for IRs which bridge two compatible nodes, we model them as translations between the nodes. Since the two models are consistent in terms of mathematical form, they can be optimized in a unified and elegant way.

Table 3.4 Statistics of the three datasets. t_u denotes the type of node u, $\langle u, r, v \rangle$ is a node-relation triple.

Datasets	Nodes	Number of Nodes	Relations ($t_u \sim t_v$)	Number of Relations	Avg. Degree of t_u	Avg. Degree of t_v	Measures $D(r)$	Measures $S(r)$	Relation Category
DBLP	Term (T)	8,811	PC	14,376	1.0	718.8	718.8	0.05	AR
	Paper (P)	14,376	APC	24,495	2.9	2089.7	720.6	0.085	AR
	Author (A)	14,475	AP	41,794	2.8	2.9	1.0	0.0002	IR
	Conference (C)	20	PT	88,683	6.2	10.7	1.7	0.0007	IR
			APT	260,605	18.0	29.6	1.6	0.002	IR
Yelp	User (U)	1,286	BR	2,614	1.0	1307.0	1307.0	0.5	AR
	Service (S)	2	BS	2,614	1.0	1307.0	1307.0	0.5	AR
	Business (B)	2,614	BL	2,614	1.0	290.4	290.4	0.1	AR
	Star Level (L)	9	UB	30,838	23.9	11.8	2.0	0.009	IR
	Reservation (R)	2	BUB	528,332	405.3	405.3	1.0	0.07	IR
AMiner	Paper (P)	127,623	PC	127,623	1.0	1263.6	1264.6	0.01	AR
	Author (A)	164,472	APC	232,659	2.2	3515.6	1598.0	0.01	AR
	Reference (R)	147,251	AP	355,072	2.2	2.8	1.3	0.00002	IR
	Conference (C)	101	PR	392,519	3.1	2.7	1.1	0.00002	IR
			APR	1,084,287	7.1	7.9	1.1	0.00004	IR

3.4.2 Preliminary

In this section, we first describe three real-world HGs and analyze the structural characteristics of relations in the HGs. Then we present two structure-related measures which can consistently distinguish various relations quantitatively.

Before analyzing the structural characteristics of relations, we first introduce three datasets used in this section, including DBLP[1], Yelp[2] and AMiner[3] [44]. The statistics of these datasets are illustrated in Table 3.4. In order to explore the structural characteristics of relations, we present mathematical analysis on the above datasets.

Since the degree of nodes can well reflect the structures of networks [51], we define a degree-based measure $D(r)$ to explore the distinction of various relations in an HG. Specifically, we compare the average degrees of two types of nodes connected with the relation r, via dividing the larger one by the smaller one ($D(r) \geq 1$). Formally, given a relation r with nodes u and v (i.e., node relation triple $\langle u, r, v \rangle$), t_u and t_v are the node types of u and v, we define $D(r)$ as follows:

$$D(r) = \frac{\max[\bar{d}_{t_u}, \bar{d}_{t_v}]}{\min[\bar{d}_{t_u}, \bar{d}_{t_v}]}, \tag{3.21}$$

where $\bar{d}_{t_u}$ and $\bar{d}_{t_v}$ are the average degrees of nodes of the types t_u and t_v respectively.

A large value of $D(r)$ indicates quite inequivalent structural roles of two types of nodes connected via the relation r (one-centered-by-another), while a small value of $D(r)$ means compatible structural roles (peer-to-peer). In other words, relations with a large value of $D(r)$ show much stronger affiliation relationships. Nodes connected via such relations share much more similar properties [8]. While relations with a small value of $D(r)$ implicate much stronger interaction relationships. Therefore, we call the two categories of relations as *Affiliation Relations* (ARs) and *Interaction Relations* (IRs), respectively.

In order to better understand the structural difference between various relations, we take the DBLP network as an example. As shown in Table 3.4, for the relation PC with $D(PC) = 718.8$, the average degree of nodes with type P is 1.0 while that of nodes with type C is 718.8. It shows that papers and conferences are structurally inequivalent. Papers are centered by conferences. While $D(AP) = 1.1$ indicates that authors and papers are compatible and peer-to-peer in structure. This is consistent with our common sense. Semantically, the relation PC means that *'papers are published in conferences'*, indicating an affiliation relationship. Differently, AP means that *'authors write papers'*, which explicitly describes an interaction relationship.

In fact, we can also define some other measures to capture the structural difference. For example, we compare the relations in terms of sparsity, which can be defined as:

---

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

[2] https://www.yelp.com/dataset/

[3] https://www.aminer.cn/citation

$$S(r) = \frac{N_r}{N_{t_u} \times N_{t_v}}, \tag{3.22}$$

where N_r represents the number of relation instances following r. N_{t_u} and N_{t_v} mean the number of nodes with type t_u and t_v, respectively. The measure can also consistently distinguish the relations into two categories: ARs and IRs. The detailed statistics of all the relations in the three HGs are shown in Table 3.4.

Evidently, Affiliation Relations and Interaction Relations exhibit rather distinct characteristics: (1) ARs indicate one-centered-by-another structures, where the average degrees of the types of end nodes are extremely different. They imply an affiliation relationship between nodes. (2) IRs describe peer-to-peer structures, where the average degrees of the types of end nodes are compatible. They suggest an interaction relationship between nodes.

3.4.3 The RHINE Model

3.4.3.1 Basic Idea

Through our exploration with thorough mathematical analysis, we find that the heterogeneous relations can be typically divided into ARs and IRs with different structural characteristics. In order to exploit their distinct characteristics, we need to specifically design different while appropriate models for the different categories of relations.

For ARs, we propose to take Euclidean distance as a metric to measure the proximity of the connected nodes in the low-dimensional space. There are two motivations behind this: (1) First of all, ARs show affiliation structures between nodes, which indicate that nodes connected via such relations share similar properties. [8, 56]. Hence, nodes connected via ARs could be directly close to each other in the vector space, which is also consistent with the optimization of Euclidean distance [5]. (2) Additionally, one goal of HG embedding is to preserve the high-order proximity. Euclidean distance can ensure that both first-order and second-order proximities are preserved as it meets the condition of the triangle inequality [16].

Different from ARs, IRs indicate strong interaction relationships between compatible nodes, which themselves contain important structural information of two nodes. Thus, we propose to explicitly model an IR as a translation between nodes in the low-dimensional vector space. Additionally, the translation based distance is consistent with the Euclidean distance in the mathematical form [2]. Therefore, they can be smoothly combined in a unified and elegant manner.

3.4.3.2 Different Models for ARs and IRs

In this subsection, we introduce two different models exploited in RHINE for ARs and IRs, respectively.

Euclidean Distance for Affiliation Relations Nodes connected via ARs share similar properties [8], therefore nodes could be directly close to each other in the vector space. We take the Euclidean distance as the proximity measure of two nodes connected by an AR.

Formally, given an affiliation node-relation triple $\langle p,s,q\rangle \in P_{AR}$ where $s \in R_{AR}$ is the relation between p and q with weight w_{pq}, the distance between p and q in the latent vector space is calculated as follows:

$$f(p,q) = w_{pq}||\mathbf{X}_p - \mathbf{X}_q||_2^2, \tag{3.23}$$

in which $\mathbf{X}_p \in \mathbb{R}^d$ and $\mathbf{X}_q \in \mathbb{R}^d$ are the embedding vectors of p and q, respectively. As $f(p,q)$ quantifies the distance between p and q in the low-dimensional vector space, we aim to minimize $f(p,q)$ to ensure that nodes connected by an AR are close to each other. Hence, we define the margin-based loss [2] function as follows:

$$\begin{aligned} L_{EuAR} = \sum_{s\in R_{AR}} \sum_{\langle p,s,q\rangle \in P_{AR}} \\ \sum_{\langle p',s,q'\rangle \in P'_{AR}} \max[0, \gamma + f(p,q) - f(p',q')], \end{aligned} \tag{3.24}$$

where $\gamma > 0$ is a margin hyperparameter. P_{AR} is the set of positive affiliation node-relation triples, while P'_{AR} is the set of negative affiliation node-relation triples.

Translation-based Distance for Interaction Relations Interaction Relations demonstrate strong interactions between nodes with compatible structural roles. Thus, different from ARs, we explicitly model IRs as translations between nodes.

Formally, given an interaction node-relation triple $\langle u,r,v\rangle$ where $r \in R_{IR}$ with weight w_{uv}, we define the score function as:

$$g(u,v) = w_{uv}||\mathbf{X}_u + \mathbf{Y}_r - \mathbf{X}_v||, \tag{3.25}$$

where $\mathbf{X}_u$ and $\mathbf{X}_v$ are the node embeddings of u and v respectively, and $\mathbf{Y}_r$ is the embedding of the relation r. Intuitively, this score function penalizes deviation of $(\mathbf{X}_u + \mathbf{Y}_r)$ from the vector $\mathbf{X}_v$.

For each interaction node-relation triple $\langle u,r,v\rangle \in P_{IR}$, we define the margin-based loss function as follows:

$$\begin{aligned} L_{TrIR} = \sum_{r\in R_{IR}} \sum_{\langle u,r,v\rangle \in P_{IR}} \\ \sum_{\langle u',r,v'\rangle \in P'_{IR}} \max[0, \gamma + g(u,v) - g(u',v')] \end{aligned} \tag{3.26}$$

where P_{IR} is the set of positive interaction node-relation triples, while P'_{IR} is the set of negative interaction node-relation triples.

3.4.3.3 A Unified Model for HG Embedding

Finally, we combine the two models for different categories of relations by minimizing the following loss function:

$$L = L_{EuAR} + L_{TrIR} \tag{3.27}$$

$$= \sum_{s \in R_{AR}} \sum_{\langle p,s,q \rangle \in P_{AR}} \sum_{\langle p',s,q' \rangle \in P'_{AR}} \max[0, \gamma + f(p,q) - f(p',q')]$$
$$+ \sum_{r \in R_{IR}} \sum_{\langle u,r,v \rangle \in P_{IR}} \sum_{\langle u',r,v' \rangle \in P'_{IR}} \max[0, \gamma + g(u,v) - g(u',v')]$$

Sampling Strategy As shown in Table 3.4, the distributions of ARs and IRs are quite unbalanced. What's more, the proportion of relations are unbalanced within ARs and IRs. Traditional edge sampling may suffer from under-sampling for relations with a small amount or over-sampling for relations with a large amount. To address the problems, we draw positive samples according to their probability distributions. As for negative samples, we follow previous work [2] to construct a set of negative node-relation triples $P'_{(u,r,v)} = \{(u',r,v)|u' \in V\} \cup \{(u,r,v')|v' \in V\}$ for the positive node-relation triple (u,r,v), where either the head or tail is replaced by a random node, but not both at the same time.

3.4.4 Experiments

3.4.4.1 Experimental Settings

Datasets As described in Subsection 3.4.2, we conduct experiments on three datasets, including DBLP, Yelp and AMiner. The statistics of them are summarized in Table 3.4.

Baselines We compare our proposed model RHINE with six state-of-the-art network embedding methods: two classic homogeneous graph embedding methods DeepWalk [28], LINE [43], and four heterogeneous graph embedding methods PTE [42], ESim [33], HIN2Vec [9] and Metapath2vec [7].

Evaluation Metrics We use different evaluation metrics on the following tasks:

- **Node Clustering** The experiments leverage K-means to cluster the nodes and evaluate the results in terms of normalized mutual information (NMI).
- **Link Prediction** We model the link prediction problem as a binary classification problem that aims to predict whether a link exists, and use Area Under Curve (AUC) and F1 score as evaluation metrics.

- **Multi-Class Classification** In this task, we employ the same labeled data used in the node clustering task. We use Micro-F1 and Macro-F1 scores as the metrics for evaluation.

Parameter Settings For a fair comparison, we set the embedding dimension $d = 100$ and the size of negative samples $k = 3$ for all models. For DeepWalk, HIN2Vec and metapath2vec, we set the number of walks per node $w = 10$, the walk length $l = 100$ and the window size $\tau = 5$. For our model RHINE, the margin γ is set to 1.

3.4.4.2 Node Clustering

Table 3.5 Performance Evaluation of Node Clustering.

Methods	DBLP	Yelp	AMiner
DeepWalk	0.3884	0.3043	0.5427
LINE-1st	0.2775	0.3103	0.3736
LINE-2nd	0.4675	0.3593	0.3862
PTE	0.3101	0.3527	0.4089
ESim	0.3449	0.2214	0.3409
HIN2Vec	0.4256	0.3657	0.3948
metapath2vec	0.6065	0.3507	0.5586
RHINE	**0.7204**	**0.3882**	**0.6024**

Table 3.6 Performance Evaluation of Link Prediction.

Methods	DBLP (A-A)		DBLP (A-C)		Yelp (U-B)		AMiner (A-A)		AMiner (A-C)	
	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1
DeepWalk	0.9131	0.8246	0.7634	0.7047	0.8476	0.6397	0.9122	0.8471	0.7701	0.7112
LINE-1st	0.8264	0.7233	0.5335	0.6436	0.5084	0.4379	0.6665	0.6274	0.7574	0.6983
LINE-2nd	0.7448	0.6741	0.8340	0.7396	0.7509	0.6809	0.5808	0.4682	0.7899	0.7177
PTE	0.8853	0.8331	0.8843	0.7720	0.8061	0.7043	0.8119	0.7319	0.8442	0.7587
ESim	0.9077	0.8129	0.7736	0.6795	0.6160	0.4051	0.8970	0.8245	0.8089	0.7392
HIN2Vec	0.9160	0.8475	0.8966	0.7892	0.8653	0.7709	0.9141	0.8566	0.8099	0.7282
metapath2vec	0.9153	0.8431	0.8987	0.8012	0.7818	0.5391	0.9111	0.8530	0.8902	0.8125
RHINE	**0.9315**	**0.8664**	**0.9148**	**0.8478**	**0.8762**	**0.7912**	**0.9316**	**0.8664**	**0.9173**	**0.8262**

As shown in Table 3.5, our model RHINE significantly outperforms all the compared methods. (1) Compared with the best competitors, the clustering performance of our model RHINE improves by 18.79%, 6.15% and 7.84% on DBLP, Yelp and AMiner, respectively. It demonstrates the effectiveness of our model RHINE by distinguishing the various relations with different structural characteristics in HGs. In addition, it also validates that we utilize appropriate models for different categories

of relations. (2) In all baseline methods, homogeneous network embedding models achieve the lowest performance, because they ignore the heterogeneity of relations and nodes. (3) RHINE significantly outperforms existing HGs embedding models (i.e., ESim, HIN2Vec and metapath2vec) on all datasets. We believe the reason is that our proposed RHINE with appropriate models for different categories of relations can better capture the structural and semantic information of HGs.

3.4.4.3 Link Prediction

The results of link prediction task are reported in Table 3.6 with respect to AUC and F1 score. It is clear that our model performs better than all baseline methods on three datasets. The reason behind the improvement is that our model based on Euclidean distance modeling relations can capture both the first-order and second-order proximities. In addition, our model RHINE distinguishes multiple types of relations into two categories in terms of their structural characteristics, and thus can learn better embeddings of nodes, which are beneficial for predicting complex relationships between two nodes.

3.4.4.4 Multi-Class Classification

We summarize the results of classification in Table 3.7. As we can observe, (1) RHINE achieves better performance than all baseline methods on all datasets except Aminer. It improves the performance of node classification by about 4% on both DBLP and Yelp averagely. In terms of AMiner, the RHINE performs slightly worse than ESim, HIN2vec and metapath2vec. This may be caused by over-capturing the information of relations *PR* and *APR* (*R* represents references). Since an author may write a paper referring to various fields, these relations may introduce some noise. (2) Although ESim and HIN2Vec can model multiple types of relations in HGs, they fail to perform well in most cases. Our model RHINE achieves good performance due to the respect of distinct characteristics of various relations.

The more detailed method description and experiment validation can be seen in [23].

3.5 Network Schema Preservation

3.5.1 Overview

Despite the success of meta-path guided HG embedding methods, the selection of meta-paths still remains an open yet challenging problem [38]. The design of meta-path schemes significantly relies on domain knowledge. Manually selecting meta-

Table 3.7 Performance Evaluation of Multi-class Classification.

Methods	DBLP		Yelp		AMiner	
	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1
DeepWalk	0.7475	0.7500	0.6723	0.7012	0.9386	0.9512
LINE-1st	0.8091	0.8250	0.4872	0.6639	0.9494	0.9569
LINE-2nd	0.7559	0.7500	0.5304	0.7377	0.9468	0.9491
PTE	0.8852	0.8750	0.5389	0.7342	0.9791	0.9847
ESim	0.8867	0.8750	0.6836	0.7399	0.9910	0.9948
HIN2Vec	0.8631	0.8500	0.6075	0.7361	**0.9962**	**0.9965**
metapath2vec	0.8976	0.9000	0.5337	0.7208	0.9934	0.9936
RHINE	**0.9344**	**0.9250**	**0.7132**	**0.7572**	0.9884	0.9807

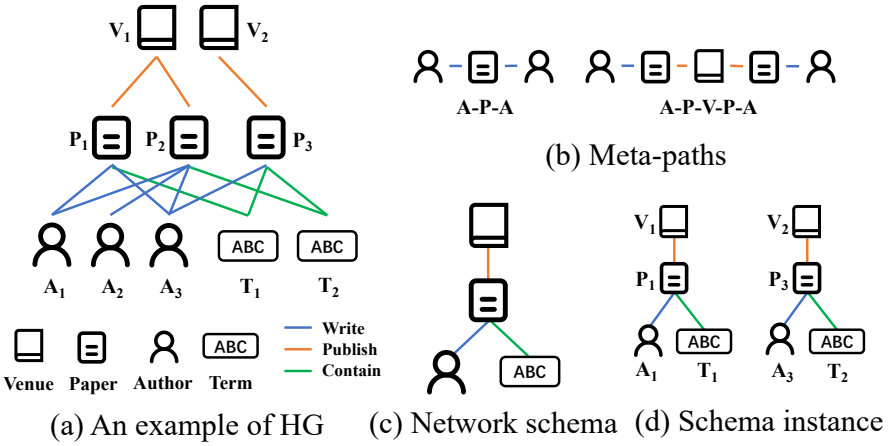


Fig. 3.7 A toy example of an HG on bibliographic data.

paths based on prior knowledge may work for a simple HG, while it is difficult to determine meta-paths for a complex HG. Furthermore, different meta-paths will result in different embeddings from different points of view, which leads to another challenging problem, i.e., how to effectively fuse different embeddings to generate uniform embeddings. Some existing works [35, 49, 22] use label information to guide the embedding fusion; unfortunately, this is not applicable in unsupervised scenarios.

To tackle the above challenges, we observe that the network schema [38], as a uniform blueprint of HG, comprehensively retains node types and their relations in an HG. Since network schema is a meta template for HG, guided by it, we can extract subgraphs (i.e., schema instances) from the HG. An example is shown in Fig. 3.7c and Fig. 3.7d, from which we can see that the schema instance depicts the high-order structure information of these four nodes, besides the first-order structure information of two nodes (i.e., pairwise structure or meta-path based structure as shown in Fig. 3.7b). Moreover, the schema instance also contains rich semantics, i.e., a schema instance (shown in Fig. 3.7d) naturally describes the overall informa-

tion, such as the author, the term, and the venue of a paper, as well as their relations. More importantly, different from meta-paths, a network schema is a unique structure for an HG, and thus we do not need domain knowledge to make a choice. These benefits of network schema motivate us to study network schema preserving HG embedding. However, it is a non-trivial task. First, *how to effectively preserve the network schema structure?* Moreover, *how to capture the heterogeneity of nodes and links inside network schema?*

In this section, we introduce a novel **N**etwork **S**chema preserving **H**eterogeneous graph **E**mbedding model named **NSHE**. Based on node embedding generated by heterogeneous graph convolutional network, NSHE optimizes the embedding via node pairs and schema instances sampled from the HG. Particularly, in the network schema preserving component, we propose a network schema sampling method, which generates sub-graphs (i.e., schema instances) naturally preserving schema structure. Furthermore, for each schema instance, a multi-task learning model is built to predict each node in the instance with other nodes, which tackles the challenge of heterogeneity.

3.5.2 The NSHE Model

3.5.2.1 Model Framework

Consider an HG $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ composed of a node set $\mathcal{V}$ and an edge set $\mathcal{E}$, along with the node type mapping function $\phi : \mathcal{V} \rightarrow \mathcal{A}$, and the edge type mapping function $\varphi : \mathcal{E} \rightarrow \mathcal{R}$, where $\mathcal{A}$ and $\mathcal{R}$ denotes the node and edge types, $|\mathcal{A}| + |\mathcal{R}| > 2$. The task is to learn the representation of nodes $\mathbf{Z} \in \mathbb{R}^{|V| \times d}$, where d is the dimension of representation.

Fig. 3.8 illustrates the framework of the proposed NSHE. NSHE preserves the pairwise and schema proximity concurrently. First, to fully exploit complex network structure and heterogeneous node feature together, we propose to learn node embedding via heterogeneous node aggregation. Second, we preserve the pairwise structure and the schema structure simultaneously. While directly performing random walk cannot generate the desired schema structure, we propose to sample schema instances and preserve the proximity inside instances. Moreover, as different types of nodes in the instances carry different context, a multi-task learning model is designed to in turn predict a target node with other context nodes to handle heterogeneity inside schema instances. Finally, NSHE iteratively updates node embeddings via optimizing the aggregation of the pairwise and schema preserving loss.

3.5.2.2 Preserving Pairwise Proximity

Despite that we need to capture the network schema structure in HG embedding, the pairwise proximity between nodes [43], as one of the most direct expressions of

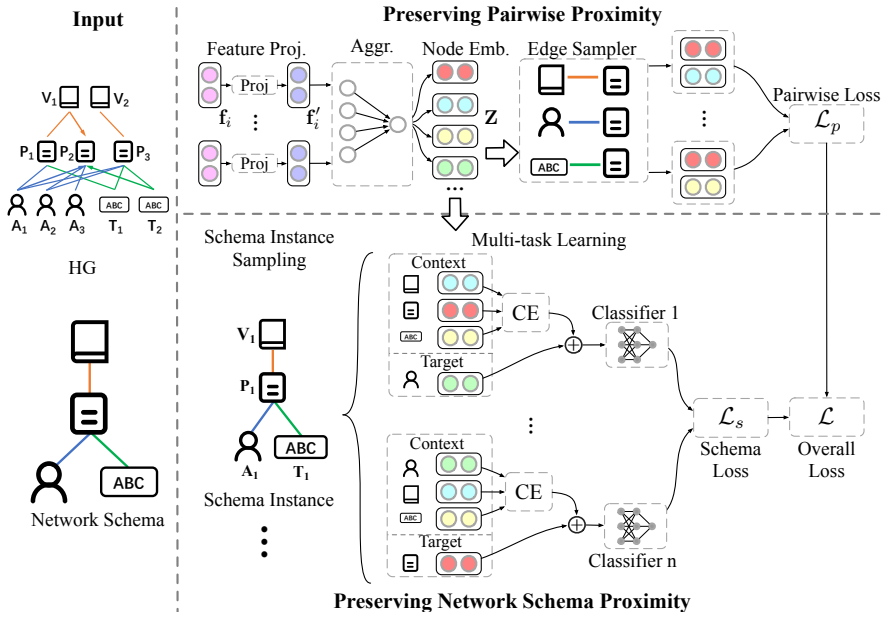


Fig. 3.8 Overview of the NSHE model.

an HG, still needs to be preserved. It demonstrates that two nodes with a link, regardless of their types, should be similar. Specifically, considering the heterogeneity of different node feature, for each node v_i with feature $\mathbf{f}_i$ and type $\phi(v_i)$, we use a type-specific mapping matrix $\mathbf{W}_{\phi(v_i)}$ to map the heterogeneous feature to a common space:

$$\mathbf{f}'_i = \sigma(\mathbf{W}_{\phi(v_i)} \cdot \mathbf{f}_i + \mathbf{b}_{\phi(v_i)}), \tag{3.28}$$

where $\sigma(\cdot)$ denotes an activation function, and $\mathbf{b}_{\phi(v_i)}$ stands for the bias vector of type $\phi(v_i)$. Based on Equation (3.28), all the nodes with different types are mapped to the common space, and we denote their mapped features as $\mathbf{H} = [\mathbf{f}'_i]$. Then, we use a L-layer graph convolutional network to generate the node embeddings [20] as:

$$\mathbf{H}^{(l+1)} = \sigma\left(\mathbf{D}^{-\frac{1}{2}}(\mathbf{A} + \mathbf{I}_{|V|})\mathbf{D}^{-\frac{1}{2}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right), \tag{3.29}$$

where $\mathbf{A}$ is the adjacency matrix, and $\mathbf{A}_{i,j} = 1$ if $(v_i, v_j) \in E$, otherwise $\mathbf{A}_{i,j} = 0$. $\mathbf{D}$ is a diagonal matrix, where $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$. $\mathbf{I}_{|V|}$ is the identity matrix of $\mathbb{R}^{|V|\times|V|}$. For the first layer, we denote $\mathbf{H}^{(0)} = \mathbf{H}$ and use the output of the L-layer graph convolutional networks as the node embedding, i.e., $\mathbf{Z} = \mathbf{H}^{(L)}$, where the i-th row of $\mathbf{Z}$ is the embedding $\mathbf{z}_{v_i}$ of node v_i.

The objective of preserving the pairwise proximity with parameters Θ can be described as:

$$\mathcal{O}_p = \arg\max_{\Theta} \prod_{v_i \in V} \prod_{v_j \in N_{v_i}} p(v_j | v_i; \Theta), \tag{3.30}$$

where $N_{v_i} = \{v_j | (v_i, v_j) \in E\}$. The conditional probability $p(v_j|v_i;\Theta)$ is defined as a softmax function:

$$p(v_j|v_i;\Theta) = \frac{\exp(\mathbf{z}_{v_j} \cdot \mathbf{z}_{v_i})}{\sum_{v_k \in V} \exp(\mathbf{z}_{v_k} \cdot \mathbf{z}_{v_i})}. \tag{3.31}$$

To calculate $p(v_j|v_i;\Theta)$ efficiently, we leverage the negative sampling method [26] and optimize Θ with the logarithm of Equation (3.30), therefore the pairwise loss $\mathcal{L}_p$ can be calculate by:

$$\begin{aligned} \mathcal{L}_p = \frac{1}{|E|} \sum_{(v_i,v_j) \in E} \big[& -\log \delta(\mathbf{z}_{v_j} \cdot \mathbf{z}_{v_i}) \\ & - \sum_{m=1}^{M_e} \mathbb{E}_{v_{j'} \sim P_n(v)} \log \delta(-\mathbf{z}_{v_{j'}} \cdot \mathbf{z}_{v_i}) \big]. \end{aligned} \tag{3.32}$$

where $\delta(x) = 1/(1+\exp(-x))$, $P_n(v)$ is the noisy distribution, and M_e is the negative edge sampling rate. Through minimizing $\mathcal{L}_p$, NSHE preserves the pairwise proximity.

3.5.2.3 Preserving Network Schema Proximity

Network Schema Instance Sampling Network schema is the blueprint of an HG [38]. Given an HG $G = (V,E)$, a network schema $T_G = (\mathcal{A},\mathcal{R})$ preserves all the node types $\mathcal{A}$ and relation types $\mathcal{R}$ inside G. Network schema proximity implies that all the nodes with different types in a network schema structure should be similar. However, as we mentioned before, the nodes in a network schema structure are usually biased, i.e., the number of nodes of a certain type is larger than those of other types. For example, in Fig. 3.7a, a paper has multiple authors, but only one venue. To alleviate such bias, we propose to sample a network schema instance defined as follows: A *network schema instance* S is the smallest sub-graph of an HG, which contains all the node types and edge types defined by the network schema T_G, if existing. By this definition, each network schema instance is composed of all the node types $\mathcal{A}$ and relation types $\mathcal{R}$ defined by the schema, i.e., one node for each type. To illustrate, Fig. 3.7d shows two instances sampled from the given HG. The sampling process is as follows: Starting from a set S with one node, we keep adding a new node to S until $|S| = |\mathcal{A}|$, where the new node satisfies: (1) its type is different from the node types in S; (2) it connects with the node(s) in S.

Schema Preserving with Multi-task Learning Now, we aim to preserve the network schema proximity by predicting whether a network schema instance exists in an HG. To this end, assume we have a network schema instance $S = \{A_1, P_1, V_1, T_1\}$ as shown in Fig. 3.8, we can predict whether A_1 exists given the set $\{P_1, V_1, T_1\}$, or whether P_1 exists given the set $\{A_1, V_1, T_1\}$, and so on. These two predictions are

different, because of the node heterogeneity. Considering this, we are motivated to design a multi-task learning model to handle the heterogeneity within schema.

Without loss of generality, assume we have the schema instance $S = \{v_i, v_j, v_k\}$, if we aim to predict whether v_i exists given $\{v_j, v_k\}$, we call v_i the **target node** and $\{v_j, v_k\}$ the **context nodes**. Therefore, each node will have two roles: one is as the target node and the other is as the context node, as well as two embeddings: target embedding and context embedding. To fully consider the heterogeneity, each node type $\phi(v_i)$ is associated with an encoder $\mathrm{CE}^{\phi(v_i)}$ to learn the context embeddings for the context nodes:

$$\mathbf{c}_{v_j} = \mathrm{CE}^{\phi(v_j)}(\mathbf{z}_{v_j}), \mathbf{c}_{v_k} = \mathrm{CE}^{\phi(v_k)}(\mathbf{z}_{v_k}), \tag{3.33}$$

where each CE stands for a fully connected layer of neural network. Then for the target node v_i, we concatenate its target embedding $\mathbf{z}_{v_i}$ with the context embeddings to obtain the schema instance embedding with target node v_i denoted as $\mathbf{z}_S^{v_i}$ as follows:

$$\mathbf{z}_S^{v_i} = \mathbf{z}_{v_i} \| \mathbf{c}_{v_j} \| \mathbf{c}_{v_k}. \tag{3.34}$$

After obtaining the embedding $\mathbf{z}_S^{v_i}$, we predict the probability of S with target node v_i, denoted as $y_S^{v_i}$, whether exists in the network:

$$y_S^{v_i} = \mathrm{MLP}^{\phi(v_i)}\left(\mathbf{z}_S^{v_i}\right), \tag{3.35}$$

where $\mathrm{MLP}^{\phi(v_i)}$ is the classifier for schema instances with target node type as $\phi(v_i)$. Similarly, when we treat v_j and v_k as the target nodes, respectively, $y_S^{v_j}$ and $y_S^{v_k}$ can also be obtained following the steps introduced above. Note that, here we take the schema instance with three nodes as an example to explain our method. However, it is easy to extend the model to schema instance with more nodes, since the process is the same.

The schema proximity loss $\mathcal{L}_s$ can be obtained by predicting the multi-tasks of the schema instances $\mathcal{S}$ sampled from HG. Additionally, to avoid trivial solutions, we also draw M_s negative examples of target type for each schema instance via replacing the target node with another node in the same type. The loss of preserving network schema can be described as:

$$\mathcal{L}_s = -\frac{1}{|\mathcal{A}||\mathcal{S}|} \sum_{S \in \mathcal{S}} \sum_{v_i \in S} \left(R_S^{v_i} \log y_S^{v_i} + \left(1 - R_S^{v_i}\right) \log\left(1 - y_S^{v_i}\right)\right), \tag{3.36}$$

where $R_S^{v_i} = 1$ if S^{v_i} is a positive network schema instance, otherwise $R_S^{v_i} = 0$. By minimizing $\mathcal{L}_s$, the schema structure is preserved.

3.5.2.4 Optimization Objective

To preserve both the pairwise proximity and the network schema proximity of HGs, NSHE optimizes the overall loss $\mathcal{L}$ by aggregating the loss of preserving pairwise proximity $\mathcal{L}_p$ and preserving schema proximity $\mathcal{L}_s$:

$$\mathcal{L} = \mathcal{L}_p + \beta \mathcal{L}_s, \tag{3.37}$$

where β is a balancing coefficient. At last, we adopt the Adam [19] algorithm to minimize the objective in Equation (3.37).

3.5.3 Experiments

3.5.3.1 Experimental Settings

Datasets In order to demonstrate the effectiveness of the proposed model, we conduct extensive experiments on three HGs datasets, including textbfDBLP [23], **IMDB** [49] and **ACM** [49].

Baselines We compare NSHE with seven state-of-the-art embedding methods including two homogeneous network embedding methods, i.e., DeepWalk [28] and LINE [43] and five heterogeneous networks embedding methods, i.e., **Metapath2Vec** [7], **HIN2Vec** [9], **HERec** [35], **DHNE** [45] and **HeGAN** [17].

Parameter Settings Here, we briefly introduce the experimental settings. For our proposed model, the feature dimension in common space and the embedding dimension d is set as 128. The negative schema instance sample rate M_s in Section 3.5.2.3 is set as 4. We perform neighborhood aggregation via an one-layer-GCN, i.e., $L = 1$, and use two-layer-MLPs for schema instance classification. For models that use meta-paths in modeling, we choose the popular meta-paths adopted in previous methods and report the best result. For models that require node feature, we apply DeepWalk [28] to generate node feature. The code and dataset is publicly available on Github[4]. The more detailed method description and experiment validation can be seen in [62].

3.5.3.2 Node Classification

In this section, we evaluate the performance of node embedding with node classification tasks. After learning the node embeddings, we train a logistic classifier with

[4] https://github.com/Andy-Border/NSHE

	DBLP-P		DBLP-A		IMDB		ACM	
	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
DeepWalk	90.12	89.45	89.44	88.48	56.52	55.24	82.17	81.82
LINE-1st	81.43	80.74	82.32	80.20	43.75	39.87	82.46	82.35
LINE-2nd	84.76	83.45	88.76	87.35	40.54	33.06	82.21	81.32
DHNE	85.71	84.67	73.30	67.61	38.99	30.53	65.27	62.31
Metapath2Vec	92.86	92.44	89.36	87.95	51.90	50.21	83.61	82.77
HIN2Vec	83.81	83.85	90.30	89.46	48.02	46.24	54.30	48.59
HERec	90.47	87.50	86.21	84.55	54.48	53.46	81.89	81.74
HeGAN	88.79	83.81	90.48	89.27	58.56	57.12	83.09	82.94
NSHE	**95.24**	**94.76**	**93.10**	**92.37**	**59.21**	**58.35**	**84.12**	**83.27**

Table 3.8 Performance evaluation of multi-class classification.

	DBLP-P	DBLP-A	IMDB	ACM
DeepWalk	46.75	66.25	0.41	**48.81**
LINE-1st	42.18	29.98	0.03	37.75
LINE-2nd	46.83	61.11	0.03	41.80
DHNE	35.33	21.00	0.05	20.25
Metapath2Vec	56.89	68.74	0.09	42.71
HIN2Vec	30.47	65.79	0.04	42.28
HERec	39.46	24.09	0.51	40.70
HeGAN	60.78	68.95	6.56	43.35
NSHE	**65.54**	**69.52**	**7.58**	44.32

Table 3.9 Performance evaluation of node clustering.

80% of the labeled nodes and use the remaining data for testing. We use Micro-F1 and Macro-F1 scores as the metrics for evaluation. The results are shown in Table 3.8, from which we have the following observations: (1) Generally speaking, HG embedding methods perform better than homogeneous network embedding methods, which proves the benefits of considering heterogeneity. (2) Though NSHE does not utilize any prior knowledge, it consistently outperforms the baselines. It demonstrates the effectiveness of our proposed method in classification tasks.

3.5.3.3 Node Clustering

We further conduct clustering tasks to evaluate the embeddings learned by NSHE. Here we utilize the K-Means model to perform node clustering and set the number of clusters for K-Means as the number of classes. The performance in terms of NMI is shown in Table 3.9. Similarly, the proposed method NSHE significantly outperforms others in most cases, which further demonstrates the effectiveness of NSHE.

The more detailed method description and experiment validation can be seen in [62].

3.6 Conclusions

The structures of heterogeneous graphs are complex and contain rich semantic information. In this chapter, we introduce four heterogeneous graph embedding methods with structure preservation. Specifically, we first introduce a meta-path based random walk method, which generates meaningful node sequences for network embedding. Then, we introduce a meta-path based collaborative filtering framework, which uses an attention mechanism to learn aspect-level network embedding. Besides, we introduce a relation-structure aware method, which distinguishes the various relations into two categories for fine-grained modelling. Finally, we introduce a meta-path independent method, which preserves network schema in network embedding. The experiments not only verify the effectiveness of these methods, but also demonstrate the important role of structural information in the embedding of heterogeneous graphs.

Some interesting future works are to explore other possible methods of distinguishing relations, paths, or network schema to better capture the structural information of heterogeneous graphs, and design more effective graph representation learning methods without the need to manually design meta-paths.

References

1. BAHDANAU, D., CHO, K., AND BENGIO, Y. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).
2. BORDES, A., USUNIER, N., GARCIA-DURAN, A., WESTON, J., AND YAKHNENKO, O. Translating embeddings for modeling multi-relational data. In *Proceedings of NIPS* (2013), pp. 2787–2795.
3. CAO, S., LU, W., AND XU, Q. Grarep: Learning graph representations with global structural information. In *CIKM* (2015), pp. 891–900.
4. CHANG, S., HAN, W., TANG, J., QI, G.-J., AGGARWAL, C. C., AND HUANG, T. S. Heterogeneous network embedding via deep architectures. In *Proceedings of SIGKDD* (2015), ACM, pp. 119–128.
5. DANIELSSON, P.-E. Euclidean distance mapping. *Computer Graphics and image processing 14*, 3 (1980), 227–248.
6. DEVLIN, J., CHANG, M.-W., LEE, K., AND TOUTANOVA, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
7. DONG, Y., CHAWLA, N. V., AND SWAMI, A. metapath2vec: Scalable representation learning for heterogeneous networks. In *KDD* (2017), pp. 135–144.
8. FAUST, K. Centrality in affiliation networks. *Social networks 19*, 2 (1997), 157–191.
9. FU, T.-Y., LEE, W.-C., AND LEI, Z. Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning. In *CIKM* (2017), pp. 1797–1806.
10. GROVER, A., AND LESKOVEC, J. node2vec: Scalable feature learning for networks. In *Proceedings of SIGKDD* (2016), ACM, pp. 855–864.
11. HAN, X., SHI, C., WANG, S., PHILIP, S. Y., AND SONG, L. Aspect-level deep collaborative filtering via heterogeneous information networks. In *IJCAI* (2018), pp. 3393–3399.
12. HARPER, F. M., AND KONSTAN, J. A. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS) 5*, 4 (2016), 19.
13. HE, R., AND MCAULEY, J. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW* (2016), pp. 507–517.

14. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., and Chua, T.-S. Neural collaborative filtering. In *WWW* (2017), pp. 173–182.
15. He, X., Zhang, H., Kan, M.-Y., and Chua, T.-S. Fast matrix factorization for online recommendation with implicit feedback. In *SIGIR* (2016), pp. 549–558.
16. Hsieh, C.-K., Yang, L., Cui, Y., Lin, T.-Y., Belongie, S., and Estrin, D. Collaborative metric learning. In *Proceedings of WWW* (2017), pp. 193–201.
17. Hu, B., Fang, Y., and Shi, C. Adversarial learning on heterogeneous information networks. In *KDD* (2019), pp. 120–129.
18. Ji, M., Han, J., and Danilevsky, M. Ranking-based classification of heterogeneous information networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining* (2011), pp. 1298–1306.
19. Kingma, D. P., and Ba, J. Adam: A method for stochastic optimization. In *ICLR (Poster)* (2015).
20. Kipf, T. N., and Welling, M. Semi-supervised classification with graph convolutional networks. In *ICLR* (2017).
21. Koren, Y., Bell, R., and Volinsky, C. Matrix factorization techniques for recommender systems. *Computer 42*, 8 (2009).
22. Linmei, H., Yang, T., Shi, C., Ji, H., and Li, X. Heterogeneous graph attention networks for semi-supervised short text classification. 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)* (2019), pp. 4823–4832.
23. Lu, Y., Shi, C., Hu, L., and Liu, Z. Relation structure-aware heterogeneous information network embedding. In *AAAI* (2019), pp. 4456–4463.
24. Ma, H., Zhou, D., Liu, C., Lyu, M. R., and King, I. Recommender systems with social regularization. In *Proceedings of the fourth ACM international conference on Web search and data mining* (2011), pp. 287–296.
25. McAuley, J., Targett, C., Shi, Q., and Van Den Hengel, A. Image-based recommendations on styles and substitutes. In *SIGIR* (2015), pp. 43–52.
26. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. In *NIPS* (2013), pp. 3111–3119.
27. Mnih, A., and Salakhutdinov, R. R. Probabilistic matrix factorization. *Advances in neural information processing systems 20* (2007), 1257–1264.
28. Perozzi, B., Al-Rfou, R., and Skiena, S. Deepwalk: Online learning of social representations. In *KDD* (2014), pp. 701–710.
29. Pham, T.-A. N., Li, X., Cong, G., and Zhang, Z. A general recommendation model for heterogeneous networks. *IEEE Transactions on Knowledge and Data Engineering 28*, 12 (2016), 3140–3153.
30. Rendle, S. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST) 3*, 3 (2012), 1–22.
31. Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme, L. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI* (2009), pp. 452–461.
32. Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. Item-based collaborative filtering recommendation algorithms. In *WWW* (2001), pp. 285–295.
33. Shang, J., Qu, M., Liu, J., Kaplan, L. M., Han, J., and Peng, J. Meta-path guided embedding for similarity search in large-scale heterogeneous information networks. *arXiv preprint arXiv:1610.09769* (2016).
34. Shi, C., Han, X., Li, S., Wang, X., Wang, S., Du, J., and Yu, P. Deep collaborative filtering with multi-aspect information in heterogeneous networks. *IEEE Transactions on Knowledge and Data Engineering* (2019).
35. Shi, C., Hu, B., Zhao, W. X., and Yu, P. S. Heterogeneous information network embedding for recommendation. *IEEE Trans. Knowl. Data Eng. 31*, 2 (2019), 357–370.
36. Shi, C., Kong, X., Huang, Y., Philip, S. Y., and Wu, B. Hetesim: A general framework for relevance measure in heterogeneous networks. *IEEE Transactions on Knowledge and Data Engineering 26*, 10 (2014), 2479–2492.

37. SHI, C., ZHANG, Z., LUO, P., YU, P. S., YUE, Y., AND WU, B. Semantic path based personalized recommendation on weighted heterogeneous information networks. In *CIKM* (2015), ACM, pp. 453–462.
38. SUN, Y., HAN, J., YAN, X., YU, P. S., AND WU, T. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *Proceedings of the VLDB Endowment 4*, 11 (2011), 992–1003.
39. SUN, Y., HAN, J., YAN, X., YU, P. S., AND WU, T. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *Proceedings of the VLDB Endowment 4*, 11 (2011), 992–1003.
40. SUN, Y., NORICK, B., HAN, J., YAN, X., PHILIP, S. Y., AND YU, X. Integrating meta-path selection with user-guided object clustering in heterogeneous information networks. In *KDD* (2012).
41. SUN, Y., YU, Y., AND HAN, J. Ranking-based clustering of heterogeneous information networks with star network schema. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining* (2009), pp. 797–806.
42. TANG, J., QU, M., AND MEI, Q. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of SIGKDD* (2015), ACM, pp. 1165–1174.
43. TANG, J., QU, M., WANG, M., ZHANG, M., YAN, J., AND MEI, Q. Line: Large-scale information network embedding. In *WWW* (2015), pp. 1067–1077.
44. TANG, J., ZHANG, J., YAO, L., LI, J., ZHANG, L., AND SU, Z. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* (2008), ACM, pp. 990–998.
45. TU, K., CUI, P., WANG, X., WANG, F., AND ZHU, W. Structural deep embedding for hyper-networks. In *AAAI* (2018), pp. 426–433.
46. VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A. N., KAISER, Ł., AND POLOSUKHIN, I. Attention is all you need. In *NIPS* (2017), pp. 5998–6008.
47. WANG, H., ZHANG, F., HOU, M., XIE, X., GUO, M., AND LIU, Q. Shine: signed heterogeneous information network embedding for sentiment link prediction. In *WSDM* (2018), ACM, pp. 592–600.
48. WANG, X., CUI, P., WANG, J., PEI, J., ZHU, W., AND YANG, S. Community preserving network embedding. In *Thirty-first AAAI conference on artificial intelligence* (2017).
49. WANG, X., JI, H., SHI, C., WANG, B., YE, Y., CUI, P., AND YU, P. S. Heterogeneous graph attention network. In *WWW* (2019), pp. 2022–2032.
50. WANG, Y., CHEN, L., CHE, Y., AND LUO, Q. Accelerating pairwise simrank estimation over static and dynamic graphs. *The VLDB Journal—The International Journal on Very Large Data Bases 28*, 1 (2019), 99–122.
51. WASSERMAN, S., AND FAUST, K. *Social network analysis: Methods and applications*, vol. 8. Cambridge university press, 1994.
52. WEI, Z., HE, X., XIAO, X., WANG, S., LIU, Y., DU, X., AND WEN, J.-R. Prsim: Sublinear time simrank computation on large power-law graphs. *arXiv preprint arXiv:1905.02354* (2019).
53. XU, K., BA, J., KIROS, R., CHO, K., COURVILLE, A., SALAKHUDINOV, R., ZEMEL, R., AND BENGIO, Y. Show, attend and tell: Neural image caption generation with visual attention. In *ICML* (2015), pp. 2048–2057.
54. XU, L., WEI, X., CAO, J., AND YU, P. S. Embedding of embedding (eoe): Joint embedding for coupled heterogeneous networks. In *Proceedings of WSDM* (2017), ACM, pp. 741–749.
55. XUE, H., DAI, X., ZHANG, J., HUANG, S., AND CHEN, J. Deep matrix factorization models for recommender systems. In *IJCAI* (2017), pp. 3203–3209.
56. YANG, J., AND LESKOVEC, J. Community-affiliation graph model for overlapping network community detection. In *Proceedings of ICDM* (2012), IEEE, pp. 1170–1175.
57. YOU, Q., JIN, H., WANG, Z., FANG, C., AND LUO, J. Image captioning with semantic attention. In *CVPR* (2016), pp. 4651–4659.
58. YU, X., REN, X., GU, Q., SUN, Y., AND HAN, J. Collaborative filtering with entity similarity regularization in heterogeneous information networks. *IJCAI HINA 27* (2013).

59. ZHANG, J., TANG, J., MA, C., TONG, H., JING, Y., AND LI, J. Panther: Fast top-k similarity search on large networks. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining* (2015), pp. 1445–1454.
60. ZHANG, J., XIA, C., ZHANG, C., CUI, L., FU, Y., AND PHILIP, S. Y. Bl-mne: Emerging heterogeneous social network embedding through broad learning with aligned autoencoder. In *Proceedings of ICDM* (2017), IEEE, pp. 605–614.
61. ZHAO, H., YAO, Q., LI, J., SONG, Y., AND LEE, D. Meta-graph based recommendation fusion over heterogeneous information networks. In *KDD* (2017), pp. 635–644.
62. ZHAO, J., WANG, X., SHI, C., LIU, Z., AND YE, Y. Network schema preserving heterogeneous information network embedding. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020* (2020), IJCAI, ijcai.org, pp. 1366–1372.
63. ZHENG, J., LIU, J., SHI, C., ZHUANG, F., LI, J., AND WU, B. Recommendation in heterogeneous information network via dual similarity regularization. *International Journal of Data Science and Analytics 3*, 1 (2017), 35–48.