# Chapter 9
# Heterogeneous Graph Representation for Industry Application

**Abstract** Heterogeneous Graph (HG) representation is closely related with the real-world applications, as heterogeneous objects and interactions are ubiquitous in many practical systems. HG representation methods deployed in real-world system should consider capturing the complex interactions among objects as well as solving the unique challenges existing in real-world systems, such as large-scale, dynamics, and multi-source information. In this chapter, we focus on summarizing the industrial level applications with HG representation. Particularly, we introduce several well deployed systems that have demonstrated the success of HG representation techniques in resolving real-world applications, including cash-out user detection, intent recommendation, share recommendation, and friend-enhanced recommendation. For industrial-level applications, we pay more attention on two key components: HG construction with industrial data and graph representation techniques on the HG.

## 9.1 Introduction

Heterogeneous objects and their relations are ubiquitous in many industrial-level applications. For example, in an E-commerce recommendation system, there are user, item, and shop objects, and the ternary interactions usually exist among these objects. However, the type information will be inevitably ignored if we utilize a homogeneous graph to model such data. Fortunately, the heterogeneous graph is a natural tool to model such complex data without information loss.

Existing methods applied in industrial applications can be roughly concluded into two categories. The first one focuses on performing subtle feature engineering from the historical user behavior data. However, this kind of method is labor-consuming. The other one is that the involved objects and their interaction are usually treated as a homogeneous graph and a homogeneous graph is adopted to learn node representation. Therefore, the heterogeneous information is largely ignored by this kind of method. But the heterogeneous information is very important for some scenarios.

In this chapter, we will introduce several successful cases which have applied HG representation methods on two categories of important industrial applications. The first category of task is the cash-out user detection which aims to predict whether a user will do cash-out transactions or not. And a novel **H**ierarchical **A**ttention mechanism based **C**ash-out **U**ser **D**etection model (named **HACUD**) is proposed to learn the users' features from the constructed HG. And the second category of task is the recommendation. We first study intent recommendation, where a novel **M**etapath-guided **E**mbedding method for **I**ntent **Rec**ommendation (named **MEIRec**) is proposed to aggregate node information through multiple meta-paths in the triple-interacted HG. Moreover, share recommendation, aiming to predict whether a user will share an item with his friend, is first studied by the **H**eterogeneous **G**raph neural network based **S**hare **Rec**ommendation model (named **HGSRec**) method. Finally, a novel friend-enhanced recommendation is studied, which multiplies the influence of friends in social recommendation. Unlike previous mentioned methods which need predefined meta-paths, we propose a novel **S**ocial **I**nfluence **A**ttentive **N**eural network (named **SIAN**), which does not require any manual selection of meta-paths. Next, we will introduce each case in detail.

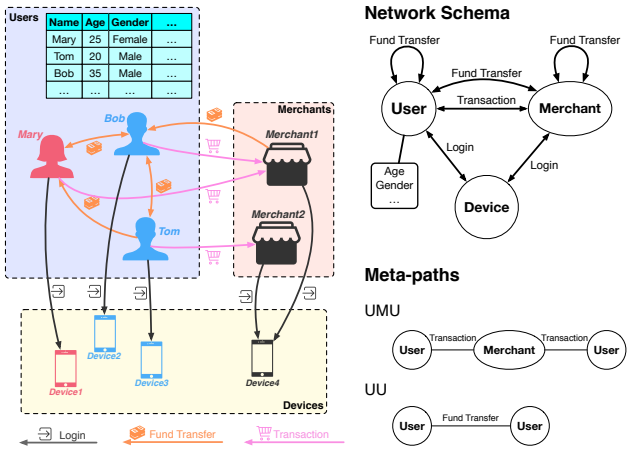
## 9.2 Cash-out User Detection

### *9.2.1 Overview*

Cash-out frauds, which are to pursue cash gains with illegal or insincere means, have seriously influenced the security of credit payment services and have become major frauds on various kinds of credit payment services. The goal of cash-out user detection is to predict whether a user will do cash-out transactions or not in the future. Thus this problem can be formulated as a binary classification problem.

Conventional solutions first perform subtle feature engineering for each user, and then a classifier, such as tree-based model or neural network, is trained based on these features. However, this kind of methods make prediction mainly based on the statistical features of a certain user, but seldom fully exploit the interaction relations between users, which may be beneficial to the cash-out user detection problem. In fact, interactions between users are important to the cash-out user detection problem. Fig. 9.1a demonstrates a general scenario of credit payment service, where there are three types of objects: users, merchants, and devices. Besides the attribute information, these objects also have rich interaction information, e.g., the fund transfer relation among users, the login relation between users and devices, and the transaction relation between users and merchants. The cash-out users not only have abnormal features, but also behave abnormally in interaction relations.

In order to tackle these problems, we propose a novel **H**ierarchical **A**ttention mechanism based **C**ash-out **U**ser **D**etection model (named **HACUD**), an HG method to predict whether a user will do cash-out transactions or not in the future. The basic




(a) Scenario of credit payment service (b) Network schema and meta-path examples

**Fig. 9.1** The AHG of the scenario of credit payment service.

idea of HACUD is to significantly enhance the feature representation of objects through fully exploiting interaction relations, i.e., with the help of meta-path based neighbors in Attributed Heterogeneous Graph (AHG). HACUD assumes that the feature representations of objects, besides intrinsic features, are also constituted by the features of their neighbors. We propose the concept of meta-path based neighbors to exploit rich structure information in AHG. Next we will explain HACUD specifically.

### *9.2.2 Preliminaries*

**Definition 1.** ***Attributed Heterogeneous Graph (AHG).*** *An AHG is denoted as* $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, X\}$ *consisting of an object set* $\mathcal{V}$*, a link set* $\mathcal{E}$ *and an attribute information matrix*[1] $X \in \mathbb{R}^{|\mathcal{V}| \times k}$*. An AHG is also associated with a node type mapping function* $\phi : \mathcal{V} \rightarrow \mathcal{A}$ *and a link type mapping function* $\psi : \mathcal{E} \rightarrow \mathcal{R}$*.* $\mathcal{A}$ *and* $\mathcal{R}$ *denote the sets of predefined object and link types, where* $|\mathcal{A}| + |\mathcal{R}| > 2$*.*

**Definition 2.** ***Meta-path based Neighbors.*** *Given a user* $u$ *and a metapath* $\rho$ *(start form* $u$*) in an AHG, the meta-path based neighbors is defined as the set of all visited objects when the object* $u$ *walks along the given metapath* $\rho$*.*

**Example 1.** *As shown in Fig. 9.1a, we construct an AHG to model the scenario of credit payment service in which cash-out fraud usually happens. It consists of multiple types of objects(i.e. User (U), Merchant (M), Device (D)) with rich attributes and*

[1] In this work, the original attributes are discretized to the same dimension.

*relations (i.e. fund transfer relation between users and transaction relation between users and merchants). Fig. 9.1b is the corresponding network schema and metapath example. In the AHG, two users can be connected via multiple meta-paths, "User-(fund transfer) -User" (UU) and "User-(transaction) -Merchant-(transaction)-User" (UMU). In addition, the meta-path based neighbor of Marry under meta-path UMU could be merchant and Bob.*

### *9.2.3 The HACUD Method*

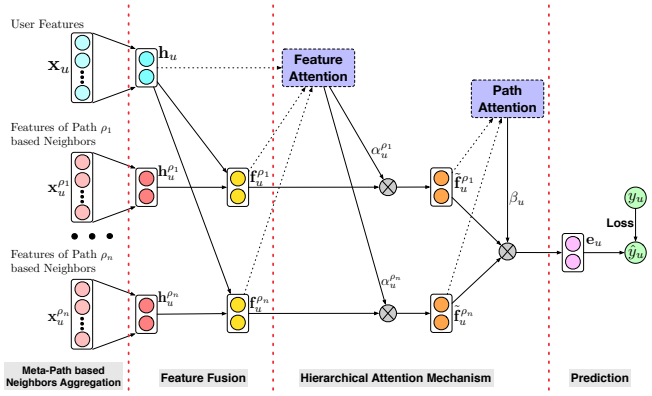



**Fig. 9.2** The architecture of the proposed model.

#### 9.2.3.1 Model Framework

We show the overall architecture of the model in Fig. 9.2. Firstly, the model aggregates neighbors for each user based on different meta-paths to integrate multiple aspects of structure information in AHG, and then transforms and fuses the original features for better representation learning. Considering that different features and meta-paths have different importances, a hierarchical attention mechanism is also used to model user preferences towards features and meta-paths.

#### 9.2.3.2 Meta-path based Neighbors Aggregation

Similar to attributed network representation [18, 36], HACUD adopts to represent a node w.r.t. a certain meta-path via aggregating features of its neighbors rather than

the one-hot representation of its neighbors. For each user $u$, the *aggregated features based on meta-path* $\rho$ can be get from the formula as below:

$$x_u^\rho = \sum_{j \in \mathcal{N}_u^\rho} w_{uj}^\rho * x_j, \tag{9.1}$$

where $\mathcal{N}_u^\rho$ is the neighbors of node $u$ based on meta-path $\rho$ and $x_j$ represents the attribute information vector associated with node $j$.

#### 9.2.3.3 Feature Fusion

For each user $u$, we can obtain its own feature $\{x_u\}$ as well as a set of its neighbor aggregation features based on multiple meta-paths $\{x_u^\rho\}_{\rho \in \mathcal{P}}$. For better representation learning, a *feature fusion* part is used to transform and fuse the original features.

Firstly, the original sparse features are projected to the low-dimensional dense representations in order to obtain the *latent representations* of user $u$ and his/her neighbors based on different meta-paths (i.e., $h_u$ and $h_u^\rho$), respectively:

$$h_u = W x_u + b, \quad h_u^\rho = W^\rho x_u^\rho + b^\rho, \tag{9.2}$$

where $W^* \in \mathbb{R}^{D \times d}$ and $b^* \in \mathbb{R}^d$ are the weight matrix and bias vector, respectively. $D$ is the dimension of original feature space[2] and $d$ is the dimension of latent representations. Next, the model fuses the latent representations of a user and his/her neighbors based on each meta-path and adds a fully-connected layer for more complicated interaction. For a meta-path $\rho$, the above procedure to get *fusional representation* $f_u^\rho$ w.r.t. meta-path $\rho$ is formulated as below,

$$f_u^\rho = \mathrm{ReLU}(W_F^\rho g(h_u, h_u^\rho) + b_F^\rho). \tag{9.3}$$

Here, $\mathbf{W}_F^\rho \in \mathbb{R}^{d \times 2d}$ and $b_F^\rho \in \mathbb{R}^d$ represent the weight matrix and bias vector based on meta-path $\rho$, respectively.

#### 9.2.3.4 Hierarchical Attention

Intuitively, different users are likely to have different preferences over the features based on different meta-paths as well as attribute information. Concretely, a user may place different importance to different-aspect features based on meta-paths. Moreover, features also have different importance for the prediction task. Therefore, hierarchical attention mechanism is applied to capture user preferences towards features and meta-paths.

**Feature Attention**. Since different features might not contribute to the prediction task equally, given the user latent representation $h_u$ and latent representation of

[2] The original attributes are discretized to sparse $D$-dimensional feature as the model input

his/her neighbors $f_u^\rho$ based on meta-path $\rho$ , a two-layer neural network is adopted to implement the attention.

$$\boldsymbol{v}_u^\rho = \text{ReLU}(W_f^1 [h_u; f_u^\rho] + b_f^1), \tag{9.4}$$

$$\boldsymbol{\alpha}_u^\rho = \text{ReLU}(W_f^2 \boldsymbol{v}_u^\rho + b_f^2), \tag{9.5}$$

where $W_f^*$ and $b_f^*$ denote the weight matrix and bias vector, respectively and $[\cdot;\cdot]$ represents the concatenation of two vectors. The following is the standard setting of neural attention networks with the softmax function.

$$\hat{\alpha}_{u,i}^\rho = \frac{\exp(\alpha_{u,i}^\rho)}{\sum_{j=1}^K \exp(\alpha_{u,j}^\rho)}. \tag{9.6}$$

Then, the final representation of user $u$ w.r.t. a meta-path $\rho$ can be computed as follows,

$$\widetilde{f}_u^\rho = \hat{\boldsymbol{\alpha}}_u^\rho \odot f_u^\rho, \tag{9.7}$$

where "$\odot$" denotes the element-wise product.

**Path Attention**. Following [23], the attention weights over different meta-paths for collaboration can be learned. Firstly, there are the attention weight of meta-path $\rho$ for user $u$ using a softmax unit as follows:

$$\beta_{u,\rho} = \frac{\exp(z^{\rho\text{T}} \cdot \widetilde{f}_u^C)}{\sum_{\rho' \in \mathcal{P}} \exp(z^{\rho'\text{T}} \cdot \widetilde{f}_u^C)}, \tag{9.8}$$

where $z^\rho \in \mathbb{R}^{|\mathcal{P}|*d}$ is the attention vector for meta-path $\rho$ and $\widetilde{f}_u^C$ is the concatenation of user $u$'s representations. After obtaining the path attention scores $\beta_{u,\rho}$, the final representation aggregating all meta-paths is given as follows:

$$e_u = \sum_{\rho \in \mathcal{P}} \beta_{u,\rho} * \widetilde{f}_u^\rho, \tag{9.9}$$

where $\widetilde{f}_u^\rho$ is the representation of neighbors for user $u$ based on meta-path $\rho$ in Eq. 9.7.

#### 9.2.3.5 Model Learning

In the end, the obtained final representation (i.e. $e_u$) are fed into multiple fully connected neural networks as follows,

$$z_u = \text{ReLU}(\mathbf{W}_L \cdots \text{ReLU}(W_1 e_u + b_1) + b_L), \tag{9.10}$$

where $W_*$ and $b_*$ respectively denote the weight matrix and the bias vector for each layer. The predicted cash-out probability is obtained via a regression layer with a

sigmoid unit:

$$p_u = \text{sigmoid}(w_p^T z_u + b_p). \tag{9.11}$$

Here $w_p$ and $b_p$ are the weight vector and the bias, respectively. The objective function is maximum likelihood estimation, which can be formulated as follows:

$$\mathcal{L}(\Theta) = \sum_{\langle u, y_u \rangle \in \mathcal{D}} (y_u \log(p_u) + (1 - y_u) \log(1 - p_u)) + \lambda ||\Theta||_2^2, \tag{9.12}$$

where $y_u$ and $p_u$ represent the ground truth and the predicted cash-out probability of user $u$, respectively. $\Theta$ is the parameter set of the proposed model and $\lambda$ is the regularizer parameter.

## *9.2.4 Experiments*

### 9.2.4.1 Experimental Settings

**Dataset.** The datasets in this section are real-world data in Ant Credit Pay. We extract two sub-datasets for the evaluation, namely Ten Days Dataset and One Month Dataset. For both datasets, the model can predict the cash-out probability of users some day in the future. In the datasets, the positive samples are users who have involved in suspected cash-out transactions within one month and the negative samples are users who have never involved in suspected cash-out transactions within one month. After preprocessing, an attributed HG based on the two datasets is constructed, consisting of 56.75 million users and 0.51 million merchants. In addition, the AHG contains 77.40 million fund transfer relations between users and 20.64 million transaction relations between users and merchants.

**Metrics.** The metric is **AUC** (i.e. Area Under the ROC Curve), a widely used metric for the performance of cash-out user detection.

**Implementation Details.** HACUD utilizes two hidden layers for prediction and randomly initializes the parameters with a xavier initializer [10]. RMSProp [27] is used as the optimizer. The batch size is set to 256, the learning rate to 0.002 and set the regularizer parameter $\lambda = 0.01$ to prevent overfitting.

### 9.2.4.2 Performance Comparison

We report the comparison results of the HACUD and baselines w.r.t. the dimension of latent representation $d$ in Table 9.1. The major findings from the experimental results can be summarized as follows:

(1) HACUD outperforms all the baselines, which indicates that the model adopts a more principled way to leverage interaction relations and attribute information for improving prediction performance.

**Table 9.1** Results of effectiveness experiments on two datasets w.r.t. the dimension of latent represantation $d$. A larger value indicates a better performance.

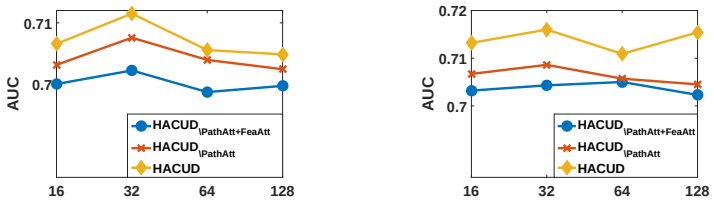
| Algorithm | AUC | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Ten Days Dataset | | | | One Month Dataset | | | |
| | $d = 16$ | $d = 32$ | $d = 64$ | $d = 128$ | $d = 16$ | $d = 32$ | $d = 64$ | $d = 128$ |
| Node2vec [11] | 0.5893 | 0.5913 | 0.5926 | 0.5930 | 0.5980 | 0.6063 | 0.6009 | 0.6021 |
| Metapath2vec [6] | 0.5914 | 0.5903 | 0.5917 | 0.5920 | 0.6005 | 0.5976 | 0.5995 | 0.5983 |
| Node2vec + Feature | 0.6455 | 0.6464 | 0.6510 | 0.6447 | 0.6541 | 0.6561 | 0.6607 | 0.6518 |
| Metapath2vec + Feature | 0.6456 | 0.6429 | 0.6469 | 0.6485 | 0.6550 | 0.6552 | 0.6523 | 0.6545 |
| Structure2vec [5] | 0.6537 | 0.6556 | 0.6598 | 0.6545 | 0.6641 | 0.6632 | 0.6657 | 0.6678 |
| GBDT [9] | 0.6389 | 0.6389 | 0.6389 | 0.6389 | 0.6467 | 0.6467 | 0.6467 | 0.6467 |
| $\text{GBDT}_{Struct}$ | 0.6948 | 0.6948 | 0.6948 | 0.6948 | 0.6968 | 0.6968 | 0.6968 | 0.6968 |
| HACUD | **0.7066** | **0.7115** | **0.7056** | **0.7049** | **0.7132** | **0.7160** | **0.7109** | **0.7154** |

(2) Among these baselines, we can find that the overall performance order is as follows: (label + attribute + structure) based methods (i.e. $\text{GBDT}_{Struct}$, Structure2vec) > (attribute + structure) based methods (i.e. Node2vec + Feature, Metapath2vec + Feature) > structure or attribute only based method (i.e. Node2vec, Metapath2vec, GBDT). It indicates that the better performances can be achieved through fusing more information. In addition, structure information (i.e. interaction relations) is really helpful for performance improvement.

(3) Compare the two variants of GBDT (i.e. traditional GBDT and $\text{GBDT}_{Struct}$), we can find that $\text{GBDT}_{Struct}$ significantly outperforms traditional GBDT and other baselines, which further demonstrates the contribution of structural features provided by meta-path based neighbors in AHG.

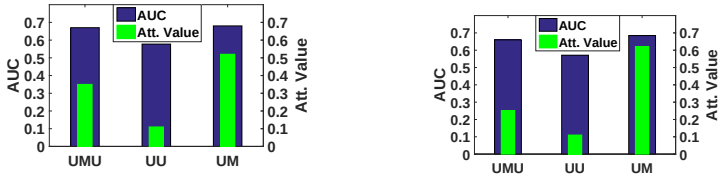
#### 9.2.4.3 Effects of Hierarchical Attention

One of the major contributions of HACUD is hierarchical attention mechanism which learns the user preference towards features and meta-paths. In order to examine its effectiveness, we compare the model with its two variants, namely $\text{HACUD}_{\backslash PathAtt}$ (HACUD without path attention) and $\text{HACUD}_{\backslash PathAtt+FeaAtt}$ (HACUD without path and feature attention). For the performance comparison in Fig. 9.3, we can find that the overall performance order is as follows: HACUD > $\text{HACUD}_{\backslash PathAtt}$ > $\text{HACUD}_{\backslash PathAtt+FeaAtt}$. The results show that the hierarchical mechanism is able to better utilize the user feature and features generated by meta-paths in two aspects. First, different meta-paths have different contributions to cash-out user prediction, which cannot be treated equally (i.e. $\text{HACUD}_{\backslash PathAtt}$). Second, each user tends to place different importance to the various attributes for each meta-path. Ignoring such influence may not be able to achieve the promising performance for fully exploiting attribute and structure information (i.e. $\text{HACUD}_{\backslash PathAtt+FeaAtt}$).




(a) Ten Days Dataset (b) One Month Dataset

**Fig. 9.3** Performance comparison of hierarchical attention w.r.t. the dimension of latent representation $d$.




(a) Ten Days Dataset (b) One Month Dataset

**Fig. 9.4** Performances comparison on different meta-paths and corresponding attention values.

#### 9.2.4.4 Impact of Different Meta-paths

Furthermore, there is another experiment about the performances based on single meta-path and corresponding average attention value in Fig. 9.4. As we have observed, the performances of HACUD with different meta-paths and the corresponding attentions are positively correlated ( i.e. important meta-paths tend to attract more attentions). In other words, the proposed HACUD model is potential to let different users focus on the proper meta-paths.

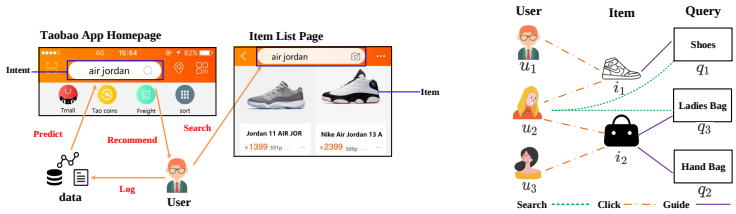
The more detailed method description and experiment validation can be seen in [12].

## 9.3 Intent Recommendation

### *9.3.1 Overview*

With the development of mobile Internet, a novel recommendation service, named intent recommendation, in many e-commerce Apps (e.g., Taobao and Amazon)

have emerged, which automatically recommends user intent (presented as several words) in a search box according to users' historical behaviors when users open an e-commerce App. Fig. 9.5a illustrates an intent recommendation example on the Taobao mobile App. According to user historic information, an intent (e.g., presented as "air jordan") will be automatically recommended in the search box when a user opens the App. If the user clicks the search button, he/she will jump to the corresponding item list page.




(a) Intent recommendation (b) Heterogeneous graph

**Fig. 9.5** Intent recommendation example on Taobao mobile application and the corresponding heterogeneous graph.

In this chapter, we define the intent recommendation as follows: automatically recommend a personalized intent for a user according to his/her historical behaviors without query input. Here, in our application scenario, intent is presented as a query, consisting of several words or terms simply and directly reflecting user intent. Existing methods for intent recommendation used in industry, such as Taobao and Amazon, usually extract handcrafted features, and then feed these features to a classifier, e.g., GBDT [9] and XGBoost [4]. These methods heavily rely on domain knowledge and need laboring feature engineering. They only utilize attribute and statistic information of users and queries, and fail to take full advantage of the rich interaction information among objects. However, the interaction information is very abundant in real systems, and it is really critical to capture user intent.

As a general information modeling method, heterogeneous graph, consisting of multiple types of objects and links, has been widely applied to many data mining tasks [25, 24, 12]. In this chapter, we firstly propose to model the intent recommendation system with an HG, through which we can flexibly exploit its rich interaction information. As shown in Fig. 9.5b, obviously, HG clearly demonstrates objects in intent recommendation (e.g., users, items and queries) and their interaction relations. Furthermore, we present a novel **M**etapath-guided **E**mbedding method for **I**ntent **Rec**ommendation (named **MEIRec**). In order to fully utilize rich interaction information in intent recommendation, we propose to learn structural feature representations of users and queries with Heterogeneous Graph Neural Network (HGNN). Concretely, we present the metapath-guided neighbours to aggregate rich neighbour information, where different aggregation functions are designed according to the

characteristics of different types of neighboring information. In addition, a uniform term embedding mechanism is designed to significantly reduce the parameter space. With the static features used in existing systems, as well as the embeddings of users and queries learned from interaction information, we build a prediction model for intent recommendation.

### *9.3.2 Problem Formulation*

**Definition 3.** ***Intent Recommendation***. *Given a set* $< \mathcal{U}, \mathcal{I}, \mathcal{Q}, \mathcal{W}, \mathcal{A}, \mathcal{B} >$, *where* $\mathcal{U} = \{u_1, \cdots, u_p\}$ *denotes the set of* $p$ *users,* $\mathcal{I} = \{i_1, \cdots, i_q\}$ *denotes the set of* $q$ *items,* $\mathcal{Q} = \{q_1, \cdots, q_r\}$ *denotes the set of* $r$ *queries,* $\mathcal{W} = \{w_1, \cdots, w_n\}$ *denotes the set of* $n$ *terms,* $\mathcal{A}$ *denotes the attributes associated with objects, and* $\mathcal{B}$ *denotes the interaction behaviors between different types of objects. In our application, a query* $q \in Q$ *or an item* $i \in I$, *is constituted by several terms* $w \in \mathcal{W}$. *The purpose of intent recommendation is to recommend the most related intent (i.e., query)* $q \in \mathcal{Q}$ *to a user* $u \in \mathcal{U}$.

**Example 2.** *Taking Fig. 9.5a for example, for a user* $u \in \mathcal{U}$, *when he refreshes the App, we can utilize information from* $\mathcal{A}$ *and* $\mathcal{B}$ *to calculate the preference score of* $u$ *for a candidate query* $q \in \mathcal{Q}$, *and recommend the query with the highest score as user intent to the user* $u$. *It is worth noting that the recommended query reflects user intent through exploiting user historical interaction information.*

### *9.3.3 The MEIRec Method*

#### 9.3.3.1 Model Framework

The basic idea of the proposed model MEIRec is to design a heterogeneous GNN for enriching the representations of users and queries. With the help of HG built from intent recommendation system, MEIRec leverages metapaths to guide the selection of different-step neighbors and designs a heterogeneous GNN to obtain the rich embeddings of users and queries. Moreover, we represent different types of objects with uniform term embedding for less parameters learning, since queries and titles of items are constituted by a small number of terms.

Fig. 9.6 shows the overall framework of MEIRec. First, we use the triple-object HG containing $< user, item, query >$ as input. Second, we use the uniform term embedding to generate the initial embeddings of items and queries. Third, we aggregate the information of metapath-guided neighbors to learn the embeddings of users and queries via heterogeneous GNN. After that, we fuse the embeddings of users and queries based on different metapaths, respectively. Finally, with the fused embeddings of users and queries, accompanying with static features of users and queries,

we predict the probability that a user will search a specific query. We illustrate these steps in detail in the following sections.

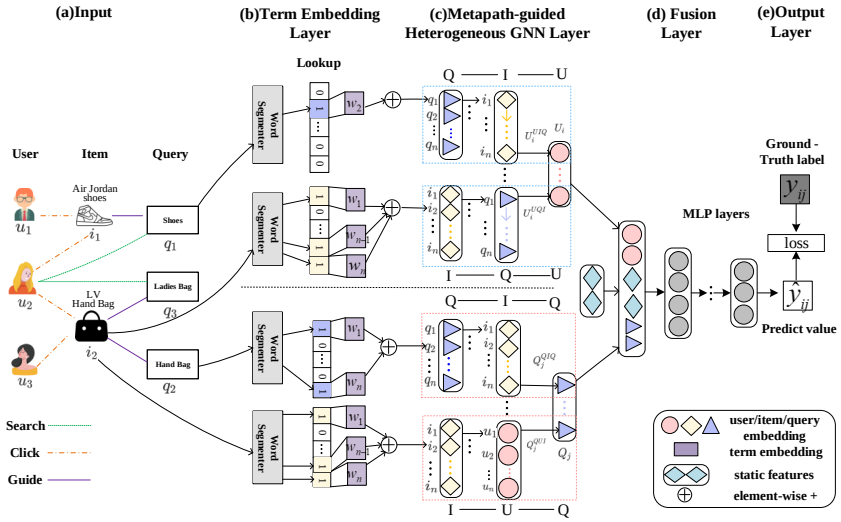



**Fig. 9.6** The framework of MEIRec.

### 9.3.3.2 Uniform Term Embedding

In previous neural-network based recommendation, every user or query should have an unique embedding. In the intent recommendation scenario, there are billions of users and queries. If we employ traditional collaborative filtering or neural-network based methods to represent all users and queries, it will make the number of parameters tremendous. Note that queries and titles of items are constituted by terms and the number of terms is not many. So we propose to represent the queries and items with a small number of term embeddings. And thus we only need to learn the term embeddings, rather than all object embeddings. This method is able to significantly reduce the number of parameters.

Specifically, we extract terms from the queries and items' titles [3], and build a term lexicon $\mathcal{W} = \{w_1, w_2, \cdots, w_{n-1}, w_n\}$. Note that queries and items (i.e., their titles) are the combination of several terms. For example, as shown in Fig 9.6a and b, query "Hand Bag" is constituted by terms "Hand" and "Bag", and item "LV Hand Bag" is constituted by terms "LV", "Hand" and "Bag". Since the number of the lexicon $\mathcal{W}$ is far less than the number of the queries and users, the uniform term embedding can

[3] Terms are important words or phrases. We use the AliWS (Alibaba Word Segmenter) to segment the queries and items' titles and select important words or phrases which contains rich meanings

significantly reduce the number of learned parameters. More importantly, the new queries that have never been searched before can be represented by these terms.

#### 9.3.3.3 Metapath-guided Heterogeneous Graph Neural Network

Inspired by the basic idea of the GCNs which generates node embeddings based on local neighbors [16, 34], we first propose a metapath-guided heterogeneous GNN. That is, we leverage metapaths to obtain different-step neighbors of an object, and the embeddings of users and queries are the aggregation of their neighbors under different metapaths.

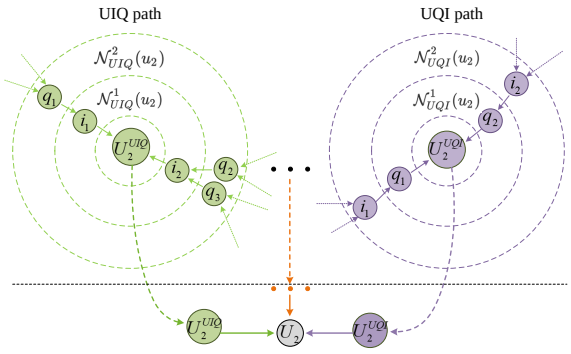



**Fig. 9.7** A toy example of metapath-guided information aggregation.

We present a toy example in Fig. 9.7 to illustrate this process. Here we describe how to obtain the embedding $U_2$ of user $u_2$ based on multiple metapaths, such as $UIQ$ and $UQI$. We first illustrate how we aggregate neighbor information along path $UIQ$. We use the uniform term embedding to obtain the initial embeddings of queries. And then we aggregate the metapath-guided neighbors to get the metapath-guided embedding of user $u_2$. According to the network structure in Fig. 2(a), we get the 1-st step neighbors set of $u_2$ , $\mathcal{N}^1_{\text{UIQ}}(u_2) = \{i_1, i_2\}$. For each node $i_k$ in the neighbors set $\mathcal{N}^1_{\text{UIQ}}(u_2)$, we extract the 2-nd step neighbor set $\mathcal{N}^2_{\text{UIQ}}(u_2) = \{q_1, q_2, q_3\}$. After we obtain the 1-st step and 2-nd step neighbors set of $u_2$, we aggregate the embeddings of 2-nd step neighbors to obtain the 1-st step neighbors' embeddings. Finally, we aggregate the embeddings of 1-st step neighbors $\{i_1, i_2\}$ to obtain embedding $U_2^{\text{UIQ}}$ of user $u_2$. Following this process, we can get different metapath-guided embeddings of $u_2$, such as $U_2^{\text{UQI}}$. Then we aggregate all the metapath-guided embeddings to get final embedding of $u_2$ (i.e., $U_2$).

#### 9.3.3.4 User Modeling

In our model, we aggregate the information of different-step neighbors to obtain the representation $U_i$ of user $u_i$ via metapath-guided heterogeneous GNN. In this section, we show how MEIRec models user embedding in detail.

As shown in the upper box in Fig 9.6c, in order to get the embedding $U_i$ of user $u_i$, we select metapaths starting from target user. We first search different-step neighbors along the metapath, and then aggregate the embeddings of neighbors step by step. Taking the metapath $UIQ$ (meaning user clicks the items which had been guided by queries) for example, we can obtain different-step neighbors of a user $u_i$. After we get the 1-st step and 2-nd step neighbors set, we aggregate the embeddings of 2-nd step neighbors (query) to obtain the 1-st step neighbors' (item) embeddings and the embedding $I_j^{\text{UIQ}}$ of item $i_j$ in $\mathcal{N}_{\text{UIQ}}^1(u_i)$ based on the metapath $UIQ$ is:

$$I_j^{\text{UIQ}} = g(E_{q_1}, E_{q_2}, \cdots), \tag{9.1}$$

where $g(\cdot)$ is the average aggregation function. And the queries $\{q_1, q_2, \cdots\}$ are the neighbors of item $i_j$.

Next, we aggregate 1-st step neighbors' (item) embeddings to obtain the embedding $U_i^{\text{UIQ}}$ of user $u_i$:

$$U_i^{\text{UIQ}} = g(I_1^{\text{UIQ}}, I_2^{\text{UIQ}}, \cdots), \tag{9.2}$$

where the items $\{i_1, i_2, \cdots\}$ are the neighbors of user $u_i$. Since users click queries or items with timestamp, we model the neighbors of users (i.e., items or queries) as a sequence data and utilize LSTM [2] to aggregate them.

Then we obtain the fused user embedding by aggregating embeddings based on different metapaths $\{\rho_1, \rho_2, \cdots, \rho_k\}$:

$$U_i = g(U_i^{\rho_1}, U_i^{\rho_2}, \cdots, U_i^{\rho_k}), \tag{9.3}$$

where the $\rho$ is metapath starting from user.

#### 9.3.3.5 Query Modeling

Similar to user information aggregation, we also obtain the fused query embedding $Q_i$ based on metapaths $\{\rho_1, \rho_2, \cdots, \rho_k\}$:

$$Q_i = g(Q_i^{\rho_1}, Q_i^{\rho_2}, \cdots, Q_i^{\rho_k}), \tag{9.4}$$

where the $\rho$ is the metapath starting from query.

#### 9.3.3.6 Optimization Objective

In our model, we predict the probability $\hat{y}_{ij}$ of user $u_i$ search the query $q_j$ which is in the range of [0,1] to ensure that the output value is a probability. Through aggregating the neighbors of user and query, we obtain the fused user embedding $U_i$ for user $u_i$ and fused query embedding $Q_j$ for query $q_j$. In addition, there are raw static features used in traditional methods, include attributes of users (queries) and static features from interaction information. We feed these static features to a Multi-Layer Perceptron for obtaining the representation of the static features $S_{ij}$. Then, we concatenate the embeddings of user, query and static features to fuse them. Finally, we feed the fused embeddings into MLP layers to get the predict score $\hat{y}_{ij}$. Then we have:

$$\hat{y}_{ij} = sigmoid(f(U_i \oplus Q_j \oplus S_{ij})), \tag{9.5}$$

where the $f(\cdot)$ is the MLP layers with only one output, $sigmoid(\cdot)$ is the sigmoid layer, and $\oplus$ is the embedding concatenate operation.

The loss function of our model is a point-wise loss function in Eq. 9.6.

$$J = \sum_{i,j \in \mathcal{Y} \cup \mathcal{Y}^-} \left( y_{ij} log \hat{y}_{ij} + (1 - y_{ij}) log(1 - \hat{y}_{ij}) \right), \tag{9.6}$$

where $y_{ij}$ is the label of the instance (i.e. 1 or 0) and the $\mathcal{Y}$ and the $\mathcal{Y}^-$ are the positive and negative instances set, respectively.

### *9.3.4 Experiments*

#### 9.3.4.1 Experimental Settings

**Dataset.** We collect a real-world large-scale dataset from Taobao mobile application from Android and IOS online. We first extract static features for user and query. And we construct an HG based on interaction data collected during 10 days. For offline experiments, we utilize the interaction data during 5 days. Specifically, each raw interaction record in the collected dataset contains $< user, query, timestamp, label >$ representing that the recommended query has been shown to user at timestamp. And the label indicates whether the user clicks the recommended query. Moreover, we use training data for different time periods (from 1 to 5 days) to predict the next one-day. Therefore, we have three datasets with different scales marked as 1-day, 3-day and 5-day. The detailed statistics of the data are shown in Table 9.2.

**Table 9.2** The statistics of the datasets.

| Dataset | 1-day | 3-day | 5-day |
|---|---|---|---|
| Training size (positive) | 2,000,000 | 6,000,000 | 9,999,999 |
| Training size (all) | 8,000,000 | 23,999,998 | 39,999,997 |
| Validation size (positive) | 2,000,000 | 2,000,000 | 1,949,143 |
| Validation size (all) | 7,999,997 | 8,000,000 | 7,949,142 |
| Train users | 4,792,621 | 11,489,531 | 16,419,735 |
| Train queries | 871,133 | 1,653,865 | 2,163,574 |
| Validation users | 4,819,489 | 4,809,497 | 4,790,912 |
| Validation queries | 876,636 | 859,488 | 787,672 |
| New users in validation set | 3,666,692 | 2,613,695 | 2,064,564 |
| Density | $4.8\times10^{-7}$ | $3.1\times10^{-7}$ | $2.8\times10^{-7}$ |

**Baselines and Evaluation Metrics.** To validate the effectiveness of our proposed model, we use the popular models used in industry (i.e., LR, DNN, and GBDT) with different feature settings and a popular neural network based model NeuMF. In particular, LR/DNN/GBDT + DW/MP means that we feed the static features of users and queries, as well as the pre-training embeddings learned by DeepWalk (DW) [21]/MetaPath2vec (MP) [6] from structural information, into LR/DNN/GBDT model. In our experiments, we use AUC [19] to evaluate the performance of different models for comparison. The large AUC value means better performance.

#### 9.3.4.2 Offline Performance Evaluation

The performances of MEIRec and the baselines are reported in Table 9.3. The major findings from the experimental results can be summarized as follows:

(1) MEIRec significantly outperforms all the compared baselines. Compared to the best performance of baselines (i.e., GBDT + MP or GBDT + DW, indicated with "*" at Table 9.3), MEIRec offers an improvement of 2.1%~4.3% in the three datasets. The results show that MEIRec achieves best results by using both static and structural features. It indicates that our model adopts a more comprehensive way to leverage static features and interaction relations for improving prediction performance.

(2) Among these baselines, we find that the order of overall performances is as follows: at the method level, GBDT > DNN > LR > NeuMF. Due to that NeuMF cannot learn the embeddings of new users and new queries appeared in the validation set, new objects' embeddings will be random variables, which makes the worst performances of NeuMF. And at the feature level, (static features + heterogeneous embeddings) based methods > (static features + homogeneous embeddings) based methods > static features based methods. This ranking indicates that fusing more information could usually get better performances. At both levels, we conclude that choosing a model plays a key role in intent recommendation, and adopting appropriate methods to fuse more information could significantly improve the performance. As a consequence, the MEIRec achieves best performances, due to the heterogeneous GNN model and utilization of rich heterogeneous interactions.

(3) As the scale of data increasing, our model outperforms the best baselines with an increased margin (from 2.1% to 4.3%). The result further confirms that our model is more scalable for large-scale datasets.

**Table 9.3** The AUC comparisons of different methods. The * indicates the best performance of the baselines. Best results of all methods are indicated in bold. The last row indicates the percentage of improvements gained by the proposed method compared to the best baseline.

| Method | 1-day | | | | 3-day | | | | 5-day | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 40% | 60% | 80% | 100% | 40% | 60% | 80% | 100% | 40% | 60% | 80% | 100% |
| NeuMF | 0.6014 | 0.6066 | 0.6136 | 0.6143 | 0.6168 | 0.6218 | 0.6249 | 0.6291 | 0.6172 | 0.6224 | 0.6246 | 0.6295 |
| LR | 0.6854 | 0.6838 | 0.6884 | 0.6889 | 0.6844 | 0.6863 | 0.6857 | 0.6865 | 0.6817 | 0.6831 | 0.6827 | 0.6836 |
| LR+DW | 0.6878 | 0.6904 | 0.6898 | 0.6930 | 0.6888 | 0.6896 | 0.6898 | 0.6900 | 0.6838 | 0.6842 | 0.6863 | 0.6867 |
| LR+MP | 0.6918 | 0.6936 | 0.6950 | 0.6969 | 0.6919 | 0.6930 | 0.6933 | 0.6933 | 0.6874 | 0.6890 | 0.6898 | 0.6899 |
| DNN | 0.6939 | 0.6981 | 0.6991 | 0.6997 | 0.6966 | 0.6985 | 0.6999 | 0.7008 | 0.6996 | 0.7011 | 0.7017 | 0.7029 |
| DNN+DW | 0.6962 | 0.6980 | 0.7003 | 0.7024 | 0.7005 | 0.7017 | 0.7024 | 0.7030 | 0.7017 | 0.7029 | 0.7040 | 0.7047 |
| DNN+MP | 0.6984 | 0.6992 | 0.7024 | 0.7057 | 0.7025 | 0.7040 | 0.7051 | 0.7057 | 0.7017 | 0.7044 | 0.7060 | 0.7069 |
| GBDT | 0.7071 | 0.7071 | 0.7067 | 0.7073 | 0.7070 | 0.7071 | 0.7072 | 0.7071 | 0.7067 | 0.7068 | 0.7072 | 0.7066 |
| GBDT+DW | 0.7114 | 0.7119 | 0.7112* | 0.7118* | 0.7109 | 0.7106 | 0.7106 | 0.7104 | 0.7109 | 0.7112 | 0.7109 | 0.7114 |
| GBDT+MP | 0.7122* | 0.7127* | 0.7110 | 0.7111 | 0.7123* | 0.7122* | 0.7122* | 0.7124* | 0.7118* | 0.7114* | 0.7114* | 0.7120* |
| MEIRec | **0.7273** | **0.7302** | **0.7339** | **0.7346** | **0.7352** | **0.7369** | **0.7380** | **0.7390** | **0.7372** | **0.7401** | **0.7409** | **0.7425** |
| Improvement | 2.1% | 2.5% | 3.2% | 3.2% | 3.2% | 3.5% | 3.6% | 3.7% | 3.6% | 4.0% | 4.1% | 4.3% |

#### 9.3.4.3 Online Experiments

To furtherly evaluate the proposed model, we conduct online experiments in Taobao mobile App. We conduct a bucket testing (i.e., A/B testing) online to test the users' response to our model against baseline. We select one bucket for baseline, and another bucket for our model. And we select the GBDT model for comparison for that GBDT is used in real system. We use the metric CTR, Unique Click[4], and UCTR to evaluate the online performance, where CTR and UCTR=Unique Click/Unique Visitor indicate change of the click ratio and visit ratio.

The results are shown in Table 9.4. We can see that, compared to the GBDT, MEIRec achieves performance improvement in all metrics, which indicates that incorporating interaction information can better capture user latent intent. Our model gains the improvement of 0.70%, 4.79% and 1.54% for Android, IOS and Total respectively in CTR. Since the CTR is to measure the ratio of clicks against impressions, the improvement of CTR shows that our model can greatly improve the user's search experience. In addition, the metric UCTR indicates how many unique visitors click the recommended query, and it gains an improvement of 2.07%, 5.43% and 2.66% for Android, IOS and Total. The improvement of UCTR shows that our model has an advantage in attracting new users to search queries.

[4] The number of visitors who performed a click

**Table 9.4** Online A/B testing experiments results.

| Data | Methods | CTR | Unique Click | UCTR |
|---|---|---|---|---|
| Android | GBDT | 1.746% | 256,116 | 13.939% |
| | MEIRec | 1.758% | 260,634 | 14.229% |
| | Improvement | 0.70% | 1.76% | 2.07% |
| IOS | GBDT | 0.7687% | 62,462 | 5.2579% |
| | MEIRec | 0.8056% | 65,895 | 5.5436% |
| | Improvement | 4.79% | 5.50% | 5.43% |
| Total | GBDT | 1.4035% | 318,578 | 10.5252% |
| | MEIRec | 1.4252% | 326,529 | 10.8052% |
| | Improvement | 1.54% | 2.50% | 2.66% |

The more detailed method description and experiment validation can be seen in [8].

## 9.4 Share Recommendation

### *9.4.1 Overview*

With the development of social e-commerce, a new recommendation paradigm, share recommendation, has sprung up recently. In particular, share recommendation aims to predict whether a user will share an item with his friend. Such recommendation demand is ubiquitous in social e-commerce. The share recommendation is significantly different from traditional recommendations, such as item recommendation [29] and friend recommendation [32]. As shown in Fig. 9.8, we can find that item recommendation aims to recommend an item to a user (i.e., essentially maximize the probability $P(i_2|u_2)$) and friend recommendation aims to recommend a friend to a user (i.e., maximize the probability $P(u_4|u_2)$). Significantly different from the above binary recommendations, the goal of share recommendation is to predict the ternary interactions among $\langle User, Item, Friend\rangle$, i.e., whether a user will share an item with his friend, maximizing the probability $P(u_3|u_2, i_3)$.

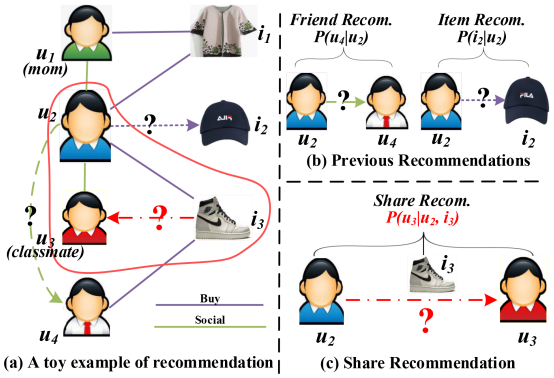



**Fig. 9.8** Share recommendation V.S. previous recommendations.

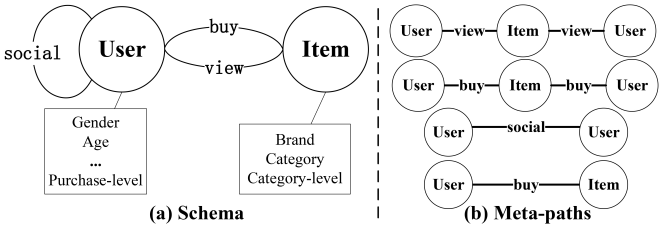



**Fig. 9.9** A typical example for share recommendation.

Deliberately considering the characteristics of share recommendation, we need to address the following challenges for modeling share recommendation. **Rich Heterogeneous Information.** Share recommendation usually contains complex heterogeneous information, including complex interactions among users and items, as well as rich feature information of users and items. **Complex ternary interaction.** Different from simple binary interaction in traditional recommendations, exemplified as $\langle u_2, i_2 \rangle$ interaction in the item recommendation and $\langle u_2, u_4 \rangle$ interaction in friend recommendation in Fig. 9.8, share recommendation faces complex ternary interaction (e.g., $\langle u_2, i_3, u_3 \rangle$ in Fig. 9.8). We need to consider the suitability of a share action, which evaluates the matching degree of three objects (e.g., $u_2$, $i_3$, $u_3$) in the share action. According to the characteristic of the recommended item, a user will recommend it to an appropriate friend, and thus how the item influence the user (or the friend) should be considered. **Asymmetric Share Action.** The share action is asymmetric and irreversible, which means the share action may not happen if we swap the roles of the user and the friend.

In this section, we first study the problem of share recommendation and propose a novel **H**eterogeneous **G**raph neural network based **S**hare **Rec**ommendation model (**HGSRec**). We model the share recommendation system as an attributed heterogeneous graph to integrate rich heterogeneous information, and then we design

HGSRec to learn the embeddings of $u$, $i$, $v$ and predict the probability of share action $\langle u,i,v\rangle$ happening. Specifically, after initializing node embedding via encoding rich node features, a tripartite heterogeneous GNNs is designed to learn the embeddings of $u$, $i$, $v$, respectively, via aggregating their meta-path based neighbors, which enables HGSRec flexibly fuse different aspects of information. Furthermore, a dual co-attention mechanism is proposed to dynamically fuse the multiple embeddings of $u$ (or $v$) under different meta-paths, considering the influence of item $i$ to user $u$ (or $v$), to improve the suitability of $\langle u,i,v\rangle$. Finally, a transitive triplet representation of $\langle u,i,v\rangle$ is employed to predict whether share action happens.

### *9.4.2 Problem Formulation*

**Definition 4.** ***Share Recommendation***. *Given an attributed heterogeneous graph $\mathcal{G}=(\mathcal{V},\mathcal{E},X)$ representing a share recommendation system, share recommendation aims to predict a share action $\langle u,i,v\rangle$ (formulated with $\langle User, Item, Friend\rangle$, or abbreviated with $\langle U,I,V\rangle$). Specifically, the purpose of share recommendation is to recommend the most likely $Friend$ $v\in\mathcal{F}(u)$ to $User$ $u\in\mathcal{V}_U$ who would like to share the $Item$ $i\in\mathcal{V}_I$ ($\langle u,i\rangle\in\mathcal{E}_O$), i.e.,* $\arg\max_v P(v|u,i)$. *The label $y_{u,i,v}\in\{0,1\}$ indicates whether share action happens.*

**Example 3.** *Fig. 9.9a shows the attributed heterogeneous graph of share recommendation. Here $u_2$ has two friends denoting as $\mathcal{F}(u_2)=\{u_1,u_3\}$. Meta-path [26], a composite relation connecting two nodes, is able to extract rich semantics. As shown in Fig. 9.9b, $User\xrightarrow{buy}Item\xleftarrow{buy}User$ (*U-b-I-b-U *for short) meaning the co-buying relations, $User\xrightarrow{social}User$ (*U-s-U *for short) meaning the social relations, $User\xrightarrow{buy}Item$ (*U-b-I *for short) meaning buy relations, and $User\xrightarrow{view}Item\xleftarrow{view}User$ (*U-v-I-v-U *for short) meaning the co-viewing relations. As shown in Fig. 9.8c, share recommendation will recommend a most likely friend, like $u_3\in\mathcal{F}(u_2)$, to a user $u_2$ who would like to share the shoes $i_3$, which essentially maximizes the probability $P(u_3|u_2,i_3)$.*

### *9.4.3 The HGSRec Method*

#### 9.4.3.1 Model Framework

The overall framework of HGSRec is shown in Fig. 9.10. Given a share action $<u,i,v>$, the basic idea of HGSRec is to learn the embeddings of $u,i,v$ to predict the probability of the action happening, with the help of delicate designs, such as tripartite heterogeneous GNNs, dual co-attention mechanism, and transitive triplet representation.

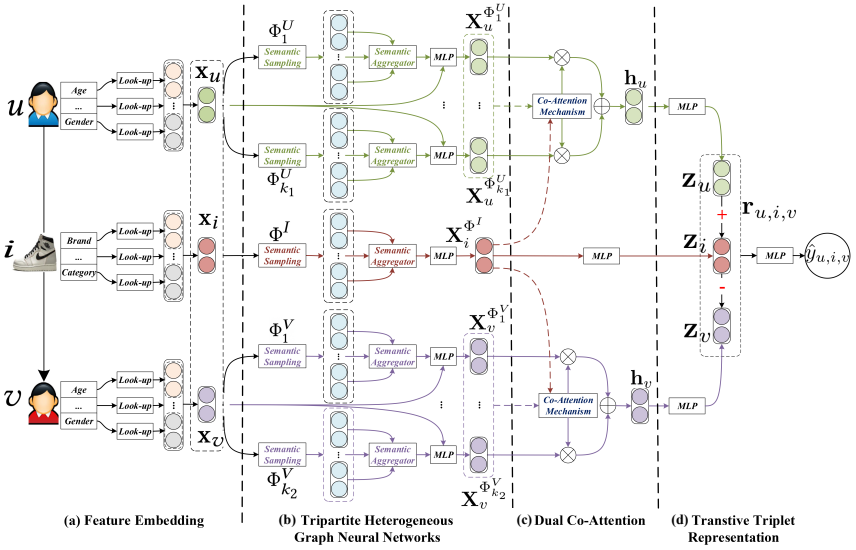



**Fig. 9.10** The overall framework of the proposed HGSRec. (a) Initializing user and item embedding via feature embedding. (b) Updating node embedding via tripartite heterogeneous graph neural networks. (c) Fusing embedding dynamicially via the dual co-attention mechanism. (d) Modeling asymmetric share action via transitive triplet representation.

### 9.4.3.2 Initialization with Feature Embedding

Firstly, we initialize node embedding via embedding their features. Different from ID embedding, feature embedding has two-fold benefits: (1) In real applications, there are numerous of newly coming nodes every day. The feature embedding effectively generates embeddings for previously unseen nodes by utilizing their features. (2) The number of features is much less than the number of nodes, which significantly reduces the number of learnable parameters.

For the $k$-th node feature $f_k \in \mathbb{R}^{|f_k|*1}$, we initialize a feature embedding matrix $\mathbf{M}^{f_k} \in \mathbb{R}^{d*|f_k|}$, where $|f_k|$ means the number of values of feature $f_k$ and $d$ is the dimension of feature embedding. The embedding of $u$'s $k$-th feature is shown as follows:

$$e_u^{f_k^U} = M^{f_k^U} \cdot u^{f_k^U}. \tag{9.1}$$

Considering all the features of user $u$, we can get the initial user embedding $x_u$, as follows:

$$x_u = \sigma\left(W_U \cdot \left(\Big\|_{k=1}^{|f^U|} e_u^{f_k^U}\right) + b_U\right), \tag{9.2}$$

where $||$ denotes the concatenation operation, $W_U$ and $b_U$ denote the weight matrix and bias vector, respectively. The same process can be done for item/friend embedding.

#### 9.4.3.3 Tripartite Heterogeneous Graph Neural Networks

Here we propose tripartite heterogeneous GNNs to learn embeddings of $u, i, v$ via corresponding heterogeneous GNN (i.e., $HeteGNN^U$, $HeteGNN^I$, and $HeteGNN^V$), respectively. Heterogeneous GNN usually follows a hierarchical manner: It first aggregates information from one kind of neighbors via one meta-path and learns the semantic-specific node embeddings in node-level. Then, it aggregates multiple semantics from different meta-paths and fuses a set of semantic-specific node embeddings in semantic-level.

Specifically, given one user $u$ and $k_1$ user-related meta-paths $\{\Phi_1^U, \Phi_2^U, \cdots, \Phi_{k_1}^U\}$, $HeteGNN^U$ is able to get $k_1$ semantic-specific user embeddings $\{x_u^{\Phi_1^U}, x_u^{\Phi_2^U}, \cdots, x_u^{\Phi_{k_1}^U}\}$.

$$x_u^{\Phi_1^U}, x_u^{\Phi_2^U}, \cdots, x_u^{\Phi_{k_1}^U} = HeteGNN^U(u). \tag{9.3}$$

Note that the number of meta-path based neighbors of different nodes could be quite different, so we need to sample fixed number of neighbors. Random sampling strategy causes heavy computation consumption and missing important nodes. Here we propose a top-$N$ semantic sampling strategy: (1) If the number of meta-path based neighbors is more than fixed number $N$, we sample top-$N$ meta-path based neighbors based on connection strength (e.g., how many times a user view an item). (2) Or else, we adopt resample to get $N$ meta-path based neighbors.

Given a user $u$ and corresponding meta-path $\Phi^U$, we propose a novel semantic aggregator $SemAgg_u^{\Phi^U}$ to aggregate sampled neighbors $\mathcal{N}_u^{\Phi^U}$ and obtain the meta-path based embedding $x_u^{\mathcal{N}_u^{\Phi^U}}$, as follows:

$$x_u^{\mathcal{N}_u^{\Phi^U}} = SemAgg_u^{\Phi^U}(\{x_n | \forall n \in \mathcal{N}_u^{\Phi^U}\}). \tag{9.4}$$

Considering the time efficiency, we adopt $MeanPooling$ to accelerate aggregating processing for faster prediction. To emphasize the property of user $u$ explicitly, we concatenate initial embedding $x_u$ and meta-path based embedding $x_u^{\mathcal{N}_u^{\Phi^U}}$ and get the semantic-specific user embedding $x_u^{\Phi^U}$,

$$x_u^{\Phi^U} = \sigma(W^{\Phi^U} \cdot (x_u || x_u^{\mathcal{N}_u^{\Phi^U}}) + b^{\Phi^U}), \tag{9.5}$$

where $W^{\Phi^U}$ and $b^{\Phi^U}$ denote the weight matrix and bias vector for meta-path $\Phi^U$, respectively. Given a set of user-related meta-paths $\{\Phi_1^U, \Phi_2^U, \cdots, \Phi_{k_1}^U\}$, we can get $k_1$ semantic-specific user embeddings $\{x_u^{\Phi_1^U}, x_u^{\Phi_2^U}, \cdots, x_u^{\Phi_{k_1}^U}\}$ which describe the characteristics of user $u$ from different aspects. The same process can be done via $HeteGNN^V$ to learn multiple semantic-specific embeddings $\{x_v^{\Phi_1^V}, x_v^{\Phi_2^V}, \cdots, x_v^{\Phi_{k_2}^V}\}$ of friend $v$. Since the characteristic of the item is much simple and stable than the user, we only adopt one meta-path $\Phi^I$ to get the embedding $x_i^{\Phi^I}$ of item $i$ via $HeteGNN^I$.

#### 9.4.3.4 Dual Co-Attention Mechanism

After obtaining a set of semantic-specific node embeddings (e.g., $\{x_u^{\Phi_1^U}, x_u^{\Phi_2^U}, \cdots, x_u^{\Phi_{k_1}^U}\}$), we aim to fuse them properly based on the complex ternary interactions $\langle u, i, v\rangle$. So a dual co-attention mechanism is designed to dynamically fuse the embeddings of $u$ (or $v$) under different meta-paths, considering the effect of item $i$, which consists of co-attention mechanism $CoAtt_{U,I}$ for $\langle U, I\rangle$ and co-attention mechanism $CoAtt_{V,I}$ for $\langle V, I\rangle$. Specifically, it learns the interaction-specific attention values of meta-paths for $\langle u, i, v\rangle$ and get the most appropriate embedding of $u, v$, with the following benefits: (1) It reinforces the dependency of $\langle u, i, v\rangle$, making HGSRec more integrated. (2) It dynamically fuses the embeddings of $u$ (or $v$), improving share suitabilities.

Taking $\langle U, I\rangle$ as an example, the co-attention mechanism $CoAtt_{U,I}$ aims to learn a set of interaction-specific co-attention weights $\{w_{u,i}^{\Phi_1^U}, w_{u,i}^{\Phi_2^U}, \cdots, w_{u,i}^{\Phi_{k_1}^U}\}$ for user $u$,

$$w_{u,i}^{\Phi_1^U}, w_{u,i}^{\Phi_2^U}, \cdots, w_{u,i}^{\Phi_{k_1}^U} = \mathrm{CoAtt}_{U,I}(x_u^{\Phi_1^U}, \cdots, x_u^{\Phi_{k1}^U}, x_i^{\Phi^I}). \tag{9.6}$$

Specifically, we concatenate the semantic-specific embedding of $u$ and $i$ and project them into co-attention space. Then, we adopt a co-attention vector $q_{U,I}$ to learn the importances of meta-paths for user $u$. The importance of meta-path $\Phi_m^U$ for $u$ in the interaction $\langle u, i\rangle$, denoted as $\alpha_{u,i}^{\Phi_m^U}$,

$$\alpha_{u,i}^{\Phi_m^U} = q_{U,I}^T \cdot \sigma(W^{U,I} \cdot (x_u^{\Phi_m^U} || x_i^{\Phi^I}) + b^{U,I}), \tag{9.7}$$

where $W^{U,I}$ and $b^{U,I}$ denote the weight matrix and bias vector, respectively. After obtaining the importances of meta-paths, we normalize them via softmax to get the co-attention weight $w_{u,i}^{\Phi_m^U}$ of meta-path $\Phi_m^U$, shown as follows:

$$w_{u,i}^{\Phi_m^U} = \frac{\exp(\alpha_{u,i}^{\Phi_m^U})}{\sum_{m=1}^{k_1} \exp(\alpha_{u,i}^{\Phi_m^U})}, \tag{9.8}$$

where $w_{u,i}^{\Phi_m^U}$ reflects the contribution of meta-path $\Phi_m^U$ in improving share suitability. With the learned weights as coefficients, we can obtain the fused embedding $h_u$ of $u$, shown as follows:

$$h_u = \sum_{m=1}^{k_1} w_{u,i}^{\Phi_m^U} \cdot x_u^{\Phi_m^U}. \tag{9.9}$$

Similar to $CoAtt_{U,I}$, $CoAtt_{V,I}$ learns a set of co-attention weights $\{w_{v,i}^{\Phi_1^V}, w_{v,i}^{\Phi_2^V}, \cdots, w_{v,i}^{\Phi_{k_2}^V}\}$ for friend $v$ and gets the fused friend embeddings $h_v$. Since we only select one meta-path for item, the fused embedding $h_i$ of item $i$ is actually $x_i^{\Phi^I}$.

#### 9.4.3.5 Transitive Triplet Representation

To predict the share action $\langle u, i, v \rangle$, we need to construct a triplet representation $r_{u,i,v}$ based on $h_u, h_i, h_v$. We first project all types of nodes in $\langle U, I, V \rangle$ into the same space via three type-specific MLPs, shown as follows:

$$z_u = MLP^U(h_u),\ z_i = MLP^I(h_i),\ z_v = MLP^V(h_v). \tag{9.10}$$

A simple way to construct the triplet representation $r_{u,i,v}$ is to concatenate all node embeddings (a.k.a., $z_u||z_i||z_v$). However, the simple concatenation cannot explicitly capture the remarkable characteristics of share action. Inspired by relational translation [1], we propose a transitive triplet representation $r_{u,i,v}$ to explicitly model the characteristics of share action via item-translating, shown as follows:

$$r_{u,i,v} = |z_u + z_i - z_v|, \tag{9.11}$$

where $|\cdot|$ denotes the absolute operation. Then, we feed $r_{u,i,v}$ into MLP and get the predict score $\hat{y}_{u,i,v}$, as follows:

$$\hat{y}_{u,i,v} = \sigma(W \cdot r_{u,i,v} + b), \tag{9.12}$$

where $W$ and $b$ denote the weight vector and bias scalar, respectively. Finally, we calculate the cross-entropy loss,

$$L = \sum_{u,i,v \in \mathcal{D}} (y_{u,i,v} \log \hat{y}_{u,i,v} + (1 - y_{u,i,v}) \log (1 - \hat{y}_{u,i,v})), \tag{9.13}$$

where $y_{u,i,v}$ is the label of the triplet, $\mathcal{D}$ denotes the dataset.

### *9.4.4 Experiments*

#### 9.4.4.1 Experimental Settings

**Dataset.** We collect data from Taobao platform, ranging from 2019/10/09 to 2019/10/14, and construct an AHG (shown in Fig. 9.9). Each sample contains a share action $\langle u, i, v \rangle$ and corresponding label $y_{u,i,v} \in \{0, 1\}$.We select four meta-paths including *U-s-U*, *U-b-I-b-U* and *U-v-I-v-U* for the user and *U-b-I* for the item. In offline experiments, we use the last day (i.e., 2019/10/14) as validation set and the previous 3/4/5 days as training sets, marked as **3-days**, **4-days**, and **5-days**, respectively. The details of the datasets are shown in Table 9.5.

**Table 9.5** The statistics of the datasets.

| Dataset | 3-days | 4-days | 5-days |
|---|---|---|---|
| #Train $\langle u, i, v \rangle$ | 3,324,367 | 4,443,996 | 5,611,531 |
| #Train $Users$ | 1,064,426 | 1,315,126 | 1,546,017 |
| #Train $Items$ | 537,048 | 679,784 | 818,290 |
| #Valid $\langle u, i, v \rangle$ | 1,401,395 | | |
| #Valid $Users$ | 539,959 | | |
| #Valid $Items$ | 247,907 | | |

**Table 9.6** The AUC comparisons of different methods.

| Model | 3-days | | | | 4-days | | | | 5-days | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 40% | 60% | 80% | 100% | 40% | 60% | 80% | 100% | 40% | 60% | 80% | 100% |
| LR | 67.56 | 67.62 | 67.26 | 67.69 | 67.58 | 67.65 | 67.68 | 67.72 | 67.62 | 67.67 | 67.72 | 67.74 |
| XGBoost | 72.04 | 72.14 | 72.13 | 72.18 | 72.08 | 72.11 | 72.15 | 72.49 | 72.72 | 72.54 | 71.78 | 72.14 |
| DNN | 71.30 | 71.20 | 71.67 | 72.03 | 71.04 | 71.33 | 71.48 | 71.80 | 70.96 | 71.12 | 71.46 | 71.51 |
| SAGE | 70.55 | 70.97 | 70.86 | 70.89 | 69.82 | 69.69 | 70.46 | 71.03 | 69.11 | 69.66 | 71.25 | 71.06 |
| IGC | 62.23 | 61.78 | 62.20 | 62.25 | 61.87 | 62.30 | 63.11 | 63.17 | 62.60 | 62.91 | 63.11 | 63.15 |
| IGC+ | 73.15 | 73.37 | 73.92 | 74.34 | 73.87 | 73.99 | 74.22 | 74.51 | 74.14 | 74.22 | 74.53 | 74.79 |
| MEIRec | 64.94 | 65.10 | 65.30 | 65.53 | 65.45 | 65.55 | 65.66 | 65.72 | 65.19 | 65.58 | 66.20 | 65.63 |
| MEIRec+ | 76.82 | 77.40 | 77.06 | 78.29 | 76.97 | 77.75 | 76.87 | 76.36 | 76.58 | 77.29 | 76.63 | 77.66 |
| HGSRec$_{\setminus att}$ | 86.63 | 86.95 | 87.16 | 87.26 | 87.00 | 87.27 | 87.31 | 87.51 | 87.11 | 87.23 | 87.34 | 87.59 |
| HGSRec$_{\setminus tra}$ | 78.17 | 79.10 | 79.50 | 79.95 | 76.40 | 79.12 | 77.09 | 79.63 | 78.22 | 78.89 | 78.83 | 81.37 |
| HGSRec | **86.84** | **87.20** | **87.36** | **87.45** | **87.05** | **87.39** | **87.43** | **87.69** | **87.27** | **87.53** | **87.72** | **87.92** |
| Impro(%). | 13.0 | 12.7 | 13.4 | 11.7 | 13.1 | 12.4 | 13.7 | 14.8 | 14.0 | 13.2 | 14.5 | 13.2 |

**Baselines.** We select feature based models (i.e., LR, DNN, and XGBoost) and GNN models (i.e., GraphSAGE, IGC, and MEIRec) as baselines. Since IGC and MEIRec cannot handle ternary recommendation, we also provide tripartite versions (i.e., IGC+ and MEIRec+) for share recommendation. To validate delicate designs in HGSRec, we also test two variants of HGSRec (HGSRec$_{\setminus att}$ and HGSRec$_{\setminus tra}$).

**Evaluation Metrics and Hyperparameter Settings.** We select AUC as the evaluation metric, RMSProp as optimizer. We uniformly set feature embedding to 8, node embedding to 128, batch size to 1024, learning rate to 0.01 and dropout rate to 0.6 for deep models. For XGBoost, we set tree depth to 6, tree number to 10. For LR, we set the L1 reg to 1. For HeteGNNs, we sample 5, 10, 2 neighbors via *U-s-U*, *U-v-I-v-U* , *U-b-I-b-U* to learn multiple user embeddings and sample 50 neighbors via *U-b-I* to learn item embedding.

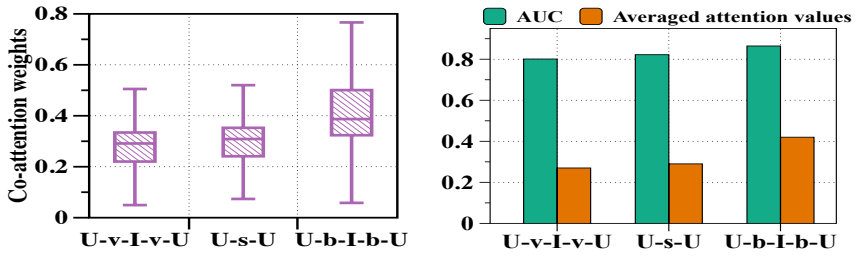
#### 9.4.4.2 Offline Performance Evaluation

As shown in Table 9.6, we have the following observations: (1) HGSRec consistently performs better than all baselines with significant improvements. Compared to the best baseline, the improvements are up to 11.7%-14.5%, indicating the su-

periority of HGSRec. (2) Most of GNNs (i.e., GraphSAGE, IGC, and MEIRec) outperform feature based methods (i.e., LR, DNN, and XGBoost), indicating the importance of structure information. When deeper insight into these methods, we can find, if employing ternary interactions, the tripartite versions (i.e., IGC+ and MEIRec+) significantly outperform the original versions. It further confirms the benefits of modeling ternary interaction for share recommendation. (3) Comparing the performance of HGSRec with its variants, we can find HGSRec achieves the best performance. The degradation of HGSRec$_{\backslash att}$ indicates the effectiveness of the dual co-attention mechanism, while the degradation of HGSRec$_{\backslash tra}$validates the superiority of transitive triplet representation. Note that the degradation of HGSRec$_{\backslash tra}$ is much more significant than that of HGSRec$_{\backslash att}$, which implies that transitive triple representation may make higher contribution than dual co-attention mechanism.

#### 9.4.4.3 Attention Analysis

The dual co-attention mechanism can dynamically fuse multiple embeddings of $User$ and $Friend$ with regard to different $Items$ and improve the share suitabilities. We first present the macro-level analysis via the box-plot figure of attention distributions over $User$ on 3-day dataset in Fig. 9.11a. Note that attention values distributions over $Friend$ also show similar phenomenons. As can be seen, the attention distributions of meta-paths are different, and the attention values of *U-b-I-b-U* is the largest with a higher variance, which illustrates that this meta-path is the most important for most users. The reason is that *U-b-I-b-U* is related to user purchasing behavior which reflects the strongest user preference. The higher variance of *U-b-I-b-U* also implies its importances varies greatly for different users. We further test HGSRec with single meta-path and show their performances with the corresponding averaged attention values in Fig 9.11b. Consistent with attention distribution, *U-b-I-b-U* is the most useful meta-path which achieves the highest AUC and gets the largest attention value.




(a) Attention distributions. (b) Performance & averaged attention values.

**Fig. 9.11** The attention analysis on 3-days dataset.

#### 9.4.4.4 Online Experiments

We deploy HGSRec on Taobao APP for online share recommendation and compare HGSRec with XGBoost via online A/B testing. Online service need to satisfy the following requirements: (1) Storage and processing for massive data. Share recommendation system is stored on MaxCompute as adjacency list for memory efficiency. (2) Abnormal share action. We filter abnormal share actions (e.g., a user shares more than thousands of items with his friend within 24 hours). (3) New feature and missing feature. New features comes everyday, so we leverage hash function to map all features, leading a slight loss of performance when hash collision happens. Missing features are padded with a specific $token$.

The online results range from 2020/01/08 to 2020/02/02 (25 days) are shown in Fig. 9.12. Here we select UCTR (UCTR=Unique Click/Unique Visitor) for online evaluation. The larger UCTR, the better performance. The long-term observations show that HGSRec consistently outperforms XGBoost with a significant gap, demonstrating the high industrial practicability and stability of HGSRec.

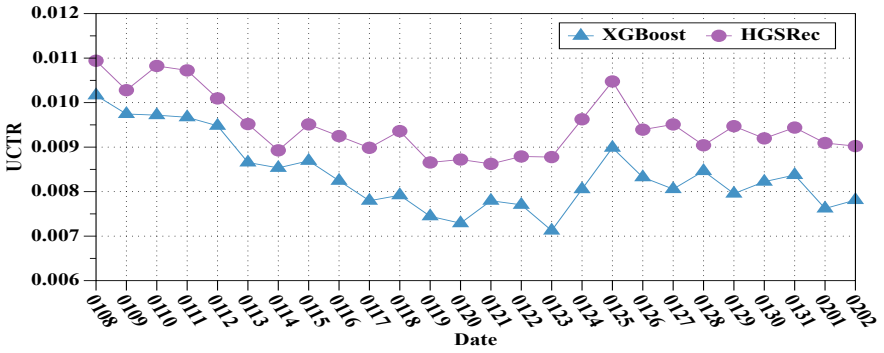



**Fig. 9.12** The results of Online A/B testing.

The more detailed method description and experiment validation can be seen in [13].

## 9.5 Friend-Enhanced Recommendation

### *9.5.1 Overview*

Nowadays, with the thriving of online social networks, people are more willing to actively express their opinions and share information with friends on social platforms. Friends become essential information sources and high-quality information filters. Impressed by the great successes of social influence in recommendation, a novel

scenario named **Friend-Enhanced Recommendation (FER)** is proposed, which multiplies the influence of friends in social recommendation. FER has two major differences from the classical social recommendation: (1) FER only recommends to the user what his/her friends have interacted with, regarding friends as high-quality information filters to provide more high-quality items. (2) All friends who have interacted with the item are explicitly displayed to the user attached to the recommended item, which highlights the critical importance of explicit social factors and improves the interpretability for user behaviors.

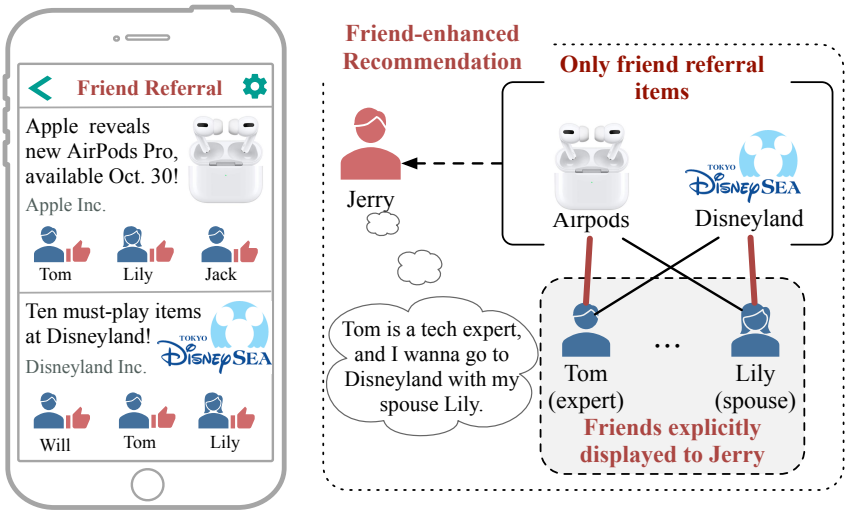



**Fig. 9.13** A typical illustration of the friend-enhanced recommendation. The left shows the scenario that *Jerry* is recommended two articles, with friends (e.g., *Tom*) who have interacted with (shared, liked, etc.) them explicitly shown underneath. The right shows the formalization of the FER problem, where only friend referral items will be recommended and friends who interacted with the item are explicitly displayed to user.

In recent years, FER systems are blooming and have been widely-used by hundreds of millions of users. Fig. 9.13 gives a typical illustration of a real-world FER. For each user-item pair, FER explicitly shows the friend set having interacted with the item, which is defined as the **Friend Referral Circle (FRC)** of the user to the item. In FER, multiple factors contribute to user clicks. The reasons for a user clicking an article may come from (1) his interests in item contents (item), (2) the recommendation of an expert (item-friend combination), or even (3) the concerns on his friends themselves (friend). In FER, users have the tendency to see *what their friends have read*, rather than to merely see *what themselves are interested in*. It could even say that social recommendation focuses on bringing social information to better recommend items, while FER aims to recommend the combination of both items and friend referrals.

As the critical characteristic of FER, the explicit FRC brings in two challenges: (1) *How to extract key information from multifaceted heterogeneous factors?* (2) *How to exploit explicit friend referral information?* To solve these issues, we propose a novel **S**ocial **I**nfluence **A**ttentive **N**eural network (named **SIAN**). Specifically, we define the FER as a user-item interaction prediction task on a heterogeneous social graph, which flexibly integrates rich information in heterogeneous objects and their interactions. First, we design an attentive feature aggregator with both node- and type-level aggregations to learn user and item representations, without being restricted to pre-defined meta-paths in some previous efforts [30, 6]. Next, we implement a social influence coupler to model the coupled influence diffusing through the explicit friend referral circles, which combines the influences of multiple factors (e.g., friends and items) with an attentive mechanism. Overall, SIAN captures valuable multifaceted factors in FER, which successfully distills the most essential preferences of users from a heterogeneous graph and friend referral circles. In experiments, SIAN significantly outperforms all competitive baselines in multiple metrics on three large, real-world datasets. Further quantitative analyses on attentive aggregation and social influence also reveal impressive sociological discoveries.

### *9.5.2 Preliminaries*

**Definition 5.** ***Heterogeneous Social Graph (HSG).*** *A heterogeneous social graph is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \mathcal{V}_U \cup \mathcal{V}_I$ and $\mathcal{E} = \mathcal{E}_F \cup \mathcal{E}_R$ are the sets of nodes and edges. Here $\mathcal{V}_U$ and $\mathcal{V}_I$ are the sets of users and items. For $u, v \in \mathcal{V}_U$, $\langle u, v\rangle \in \mathcal{E}_F$ represents the friendship between users. For $u \in \mathcal{V}_U$ and $i \in \mathcal{V}_I$, $\langle u, i\rangle \in \mathcal{E}_R$ is the interaction relation between $u$ and $i$.*

**Example 4.** *Fig. 9.13 shows an HSG containing three types of nodes, i.e., {User, Article, Media}, and multiple relations, e.g., {User-User, User-Article, User-Media, Article-Media}.*

**Definition 6.** ***Friend Referral Circle (FRC).*** *Given an HSG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we define the friend referral circle of a user $u$ w.r.t. a non-interacting item $i$ (i.e., $\langle u, i\rangle \notin \mathcal{E}_R$) as $\mathcal{C}_u(i) = \{v | \langle u, v\rangle \in \mathcal{E}_F \cap \langle v, i\rangle \in \mathcal{E}_R\}$. Here $v$ is called an **influential friend** of user $u$.*

**Example 5.** *Taking Fig. 9.13 as an example, the friend referral circle of Jerry w.r.t. the non-interacting article about AirPods is {Tom, Lily, Jack}, while the FRC in terms of the article about Disneyland is $\mathcal{C}_{Jerry}(Disneyland)$ = {Will, Tom, Lily}.*

**Definition 7.** ***Friend-Enhanced Recommendation (FER).*** *Given an HSG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and the FRC $\mathcal{C}_u(i)$ of a user $u$ w.r.t. a non-interacting item $i$, the FER aims to predict whether user $u$ has a potential preference to item $i$. That is, a prediction function $\hat{y}_{ui} = \mathcal{F}(\mathcal{G}, \mathcal{C}_u(i); \Theta)$ is to be learned, where $\hat{y}_{ui}$ is the probability that user $u$ will interact with item $i$, and $\Theta$ is the model parameters.*

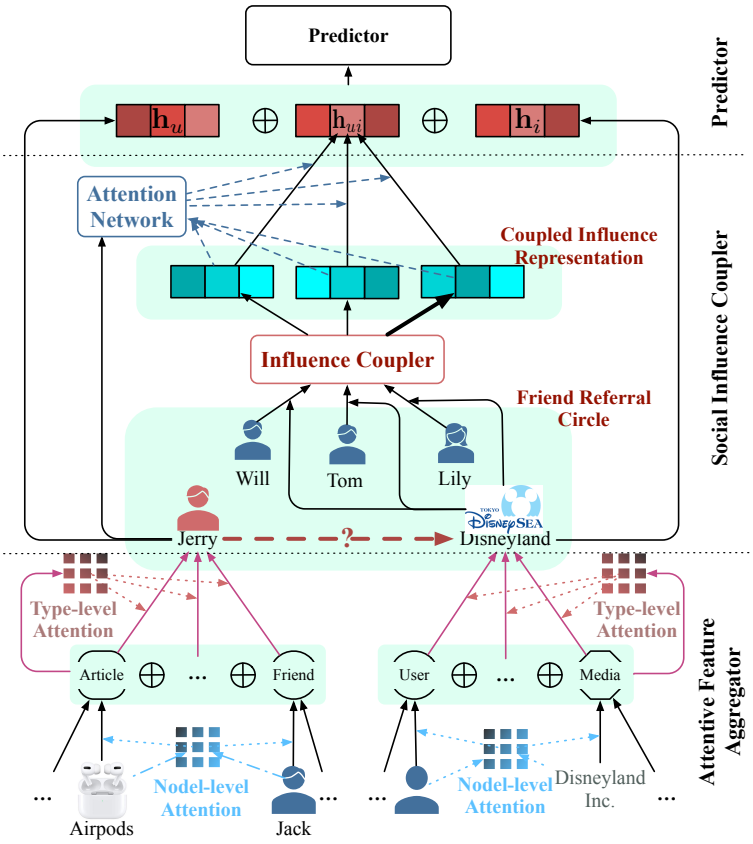



**Fig. 9.14** The overall architecture of SIAN. The attentive feature aggregator hierarchically aggregates heterogeneous neighbour features with node- and type-level attention, and outputs the representations of users and items (i.e., $h_u$ and $h_i$). The social influence coupler couples the influence of each influential friends and the item, to encode the explicit social influence into the representation (i.e., $h_{ui}$).

## *9.5.3 The SIAN Model*

### 9.5.3.1 Model Framework

As illustrated in Fig. 9.14, SIAN models the FER with an HSG. In addition to the user and item representations (e.g., $h_u$ for *Jerry* and $h_i$ for the *Disneyland* article), SIAN learns a social influence representation (e.g., $h_{ui}$) by coupling each influential friend (e.g., *Tom*) with the item. They are jointly responsible for predicting the probability $\hat{y}_{ui}$ of interaction between user $u$ and item $i$. First, each user or item node is equipped with an attentive feature aggregator with node- and type-level aggregations, which is designed to exploit multifaceted information. Second, the

influence from an influential friend (e.g., *Tom*) and an item (e.g., the *Disneyland* article) is jointly captured with a social influence coupler, which quantifies the degree of their coupled influence.

#### 9.5.3.2 Attentive Feature Aggregator

Given an HSG $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, attentive feature aggregator aims to learn user and item representations (i.e., $h_u$ and $h_i$, $u, i \in \mathcal{V}$). Considering that different neighbours of the same type might not equally contribute to the feature aggregation, and different types entail multifaceted information, we design a hierarchical node- and type-level attentive aggregation. Node-level aggregation separately models user/item features in a fine-grained manner, while type-level aggregations capture heterogeneous information.

**Node-level Attentive Aggregation.** Formally, given a user $u$, let $\mathcal{N}_u = \mathcal{N}_u^{t_1} \cup \mathcal{N}_u^{t_2} \cup \cdots \cup \mathcal{N}_u^{t_{|\mathcal{T}|}}$ denotes his/her neighbours, which is a union of $|\mathcal{T}|$ types of neighbour sets. For neighbours of type $t \in \mathcal{T}$ (i.e., $\mathcal{N}_u^t$), we represent the aggregation in the $t$ type space as the following function:

$$p_u^t = \text{ReLU}(W_p(\sum\nolimits_{k \in \mathcal{N}_u^t} \alpha_{ku} x_k) + b_p), \tag{9.1}$$

where $p_u^t \in \mathbb{R}^d$ is the aggregated embeddings of user $u$ in $t$ type space. $x_k \in \mathbb{R}^d$ is the initial embedding of the neighbour $k$, which is randomly initialized. Here $W_p \in \mathbb{R}^{d \times d}$ and $b_p \in \mathbb{R}^d$ are the weight and bias of a neural network. $\alpha_{ku}$ is the attentive contribution of neighbour $k$ to the feature aggregation of $u$,

$$\alpha_{ku} = \frac{\exp(f([x_k \oplus x_u]))}{\sum_{k' \in \mathcal{N}_u^t} \exp(f([x_{k'} \oplus x_u]))}, \tag{9.2}$$

where $f(\cdot)$ is a two-layer neural network activated with ReLu function and $\oplus$ denotes the concatenation operation. Obviously, the larger $\alpha_{ku}$, the greater contribution of neighbour $k$ to the feature aggregation of user $u$.

Given multiple types of neighbours, we can get multiple embeddings for $u$ in various type spaces, denoted as $\{p_u^{t_1}, \cdots, p_u^{t_{|\mathcal{T}|}}\}$.

**Type-level Attentive Aggregation.** Intuitively, different types of neighbours indicate various aspects of information and a node is likely to have different preferences for multiple aspects. Given a user $u$ and his/her node-level aggregated embeddings in different type spaces, we aggregate them as follows:

$$h_u = \text{ReLU}(W_h \sum\nolimits_{t \in \mathcal{T}} \beta_{tu} p_u^t + b_h), \tag{9.3}$$

where $h_u \in \mathbb{R}^d$ is the latent representation of user $u$. $\{W_h \in \mathbb{R}^{d \times d}, b_h \in \mathbb{R}^d\}$ are parameters of a neural network. $\beta_{tu}$ is the attentive preferences of type $t$ w.r.t. the feature aggregation of user $u$, as various types of neighbours contain multifaceted information and are expected to collaborate with each other. For user $u$, we concate-

nate the aggregated representations of all neighbour types, and define the following weight:

$$\beta_{tu} = \frac{\exp(a_t^\top [p_u^{t_1} \oplus p_u^{t_2} \oplus \cdots \oplus p_u^{t_{|\mathcal{T}|}}])}{\sum_{t' \in \mathcal{T}} \exp(a_{t'}^\top [p_u^{t_1} \oplus p_u^{t_2} \oplus \cdots \oplus p_u^{t_{|\mathcal{T}|}}])}, \tag{9.4}$$

where $a_t \in \mathbb{R}^{|\mathcal{T}|d}$ is a type-aware attention vector shared by all users. With Eq. 9.4, the concatenation of various neighbour types captures multifaceted information for a user, and $a_t$ encodes the global preference of each type.

Similarly, for each item $i$, the attentive feature aggregator takes the neighbours of $i$ as input, and outputs the latent representation of $i$, denoted as $h_i$.

#### 9.5.3.3 Social Influence Coupler

To exploit the FRCs and capture the effects of influential friends, we propose a social influence coupler. The different impact of the influential friends and the item on social behaviors is first coupled together, and then we attentively represent the overall influence in the FRC.

**Coupled Influence Representation.** Following [14], human behaviors are affected by various factors. In FER, whether $u$ interacts with $i$ is not simply driven by only the item itself or only the friends. More likely, the co-occurrence of friends and the item have a significant impact. As in the previous example (Fig. 9.13), when it is technology-related, the coupling between the expert (e.g. *Tom*) and the item (e.g. *AirPods*) has a greater impact than the coupling between the spouse and a tech-item, but the opposite scenario may happen for entertainment-related items. Hence, given user $u$, item $i$, and the FRC $\mathcal{C}_u(i)$, we couple the influence of each friend $v \in \mathcal{C}_u(i)$ and item $i$ as following:

$$c_{\langle v,i \rangle} = \sigma(W_c \phi(h_v, h_i) + b_c), \tag{9.5}$$

where $h_v$ and $h_i$ are aggregated representations of user $v$ and item $i$. $\phi(\cdot,\cdot)$ serves as a fusion function, which can be element-wise product or concatenation (here we adopt concatenation). $\sigma$ is the ReLU function. Obviously, Eq. 9.5 couples the features of item $i$ and the influential friend $v$, capturing the influence of both.

**Attentive Influence Degree.** With the coupled influence representation $c_{\langle v,i \rangle}$, our next goal is to obtain the influence degree of $c_{\langle v,i \rangle}$ on the user $u$. Since the influence score depends on user $u$, we incorporate the representation of user $u$ (i.e., $h_u$) into the influence score calculation with a two-layer neural network parameterized by $\{W_1, W_2, b_1, b_2\}$:

$$d'_{u \leftarrow \langle v,i \rangle} = \sigma(W_2(\sigma(W_1 \phi(c_{v,i}, h_u) + b_1)) + b_2). \tag{9.6}$$

Then, the attentive influence degree is obtained by normalizing $d'_{u \leftarrow \langle v,i \rangle}$, which can be interpreted as the impact of the influential friend $v$ on the user behavior:

$$d_{u\leftarrow\langle v,i\rangle} = \frac{\exp(d'_{u\leftarrow\langle v,i\rangle})}{\sum_{v'\in\mathcal{C}_u(i)}\exp(d'_{u\leftarrow\langle v',i\rangle})}. \tag{9.7}$$

Since the influences of friends propagate from the FRC, we attentively sum the coupled influences of the influential friends and item $v$ on user $u$:

$$h_{ui} = \sum\nolimits_{v\in\mathcal{C}_u(i)} d_{u\leftarrow\langle v,i\rangle} c_{\langle v,i\rangle}. \tag{9.8}$$

As the coupled influence representation $c_{\langle v,i\rangle}$ incorporates the latent factors of the influential friend and the item, Eq. 9.8 guarantees that the social influence propagating among them can be encoded into the latent representation $h_{ui}$.

#### 9.5.3.4 Behavior Prediction and Model Learning

With the representations of user, item and the coupled influence (i.e., $h_u$, $h_i$ and $h_{ui}$), we concatenate them and then feed it into a two-layer neural network:

$$h_o = \sigma(W_{o_2}(\sigma(W_{o_1}([h_u \oplus h_{ui} \oplus h_i]) + b_{o_1}) + b_{o_2}). \tag{9.9}$$

Then, the predicted probability of a user-item pair is obtained via a regression layer with a weight vector $w_y$ and bias $b_y$:

$$\hat{y}_{ui} = \text{sigmoid}(w_y^\top h_o + b_y). \tag{9.10}$$

Finally, to estimate model parameters $\Theta$ of SIAN, we optimize the following cross-entropy loss, where $y_{ui}$ is the ground truth and $\lambda$ is the L2-regularization parameter for reducing overfitting:

$$-\sum_{\langle u,i\rangle\in\mathcal{E}_R} (y_{ui}\log\hat{y}_{ui} + (1-y_{ui})\log(1-\hat{y}_{ui})) + \lambda||\Theta||_2^2. \tag{9.11}$$

### *9.5.4 Experiments*

#### 9.5.4.1 Experimental Settings

**Datasets.** Yelp and Douban are classical open datasets widely used in recommendation, for which we build FRCs for each user-item pair to simulate the FER scenarios. FWD is extracted from a deployed live FER system with real FRCs displayed to users. The detailed statistics of datasets are shown in Table 9.7.

**Table 9.7** Statistics of datasets.

| Datasets | Nodes | #Nodes | Relations | #Relations |
|---|---|---|---|---|
| Yelp | User (U)<br>Item (I) | 8,163<br>7,900 | User-User<br>User-Item | 92,248<br>36,571 |
| Douban | User (U)<br>Book (B) | 12,748<br>13,342 | User-User<br>User-Book | 169,150<br>224,175 |
| FWD | User (U)<br>Article (A)<br>Media (M) | 72,371<br>22,218<br>218,887 | User-User<br>User-Article<br>User-Media<br>Article-Media | 8,639,884<br>2,465,675<br>1,368,868<br>22,218 |

- **Yelp**[5] is a business review dataset containing both interactions and social relations. We first sample a set of users. For each user $u$, we construct a set of FRCs based on the given user-user relations and user-item interactions.
- **Douban**[6] is a social network related to sharing books, which including friendships between users and interaction records between users and items. As pre-processes done for Yelp, we construct a set of FRCs based on the given user-user relations and user-item interactions.
- **Friends Watching Data (FWD)** is extracted from a real-world live FER system named WeChat Top Stories after data masking, where FRCs are explicitly displayed. Based on FWD, we construct an HSG containing nearly 313 thousand nodes and 12 million edges.

**Baselines.** We compare the proposed SIAN against four types of methods, including feature/structure-based methods (i.e., MLP, DeepWal [22], node2vec [11] and metapath2vec [7]), fusion of feature/structure-based methods (i.e., DeepWalk+fea, node2vec+fea and metapath2vec+fea), graph neural network methods (i.e., GCN [17], GAT [28] and HAN [31]) and social recommendation methods (i.e., TrustMF [35] and DiffNet [33]).

**Hyperparameters Settings.** For each dataset, the ratio of training, validation and test set is 7:1:2. We adopt Adam optimizer [15] with the PyTorch implementation. The learning rate, batch size, and regularization parameter are set to 0.001, 1,024 and 0.0005 using grid search [3], determined by optimizing AUC on the validation set. For random walk based baselines, we set the walk number, walk length and window size as 10, 50, and 5, respectively. For graph neural network based methods, the number of layers is set to 2. For DiffNet, we set the regularization parameter as 0.001. The depth parameter is set to 2 as recommended in [33]. For other parameters of baselines, we optimize them empirically under the guidance of literature. Finally, for all methods except MLP, we set the size of feature vector as 64 and report performances under different embedding dimensions $\{32, 64\}$.

[5] https://www.yelp.com/dataset/challenge

[6] https://book.douban.com

**Table 9.8** Results on three datasets. The best method is bolded, and the second best is underlined. * indicate the significance level of 0.01.

| Dataset | Model | AUC | | F1 | | Accuracy | |
|---|---|---|---|---|---|---|---|
| | | $d$=32 | $d$=64 | $d$=32 | $d$=64 | $d$=32 | $d$=64 |
| Yelp | MLP | 0.6704 | 0.6876 | 0.6001 | 0.6209 | 0.6589 | 0.6795 |
| | DeepWalk | 0.7693 | 0.7964 | 0.6024 | 0.6393 | 0.7001 | 0.7264 |
| | node2vec | 0.7903 | 0.8026 | 0.6287 | 0.6531 | 0.7102 | 0.7342 |
| | metapath2vec | 0.8194 | 0.8346 | 0.6309 | 0.6539 | 0.7076 | 0.7399 |
| | DeepWalk+fea | 0.7899 | 0.8067 | 0.6096 | 0.6391 | 0.7493 | 0.7629 |
| | node2vec+fea | 0.8011 | 0.8116 | 0.6634 | 0.6871 | 0.7215 | 0.7442 |
| | metapath2vec+fea | 0.8301 | 0.8427 | 0.6621 | 0.6804 | 0.7611 | 0.7856 |
| | GCN | 0.8022 | 0.8251 | 0.6779 | 0.6922 | 0.7602 | 0.7882 |
| | GAT | 0.8076 | 0.8456 | 0.6735 | 0.6945 | 0.7783 | 0.7934 |
| | HAN | 0.8218 | 0.8476 | 0.7003 | 0.7312 | 0.7893 | 0.8102 |
| | TrustMF | 0.8183 | 0.8301 | 0.6823 | 0.7093 | 0.7931 | 0.8027 |
| | DiffNet | 0.8793 | 0.8929 | 0.8724 | 0.8923 | 0.8698 | 0.8905 |
| | SIAN | **0.9486*** | **0.9571*** | **0.8976*** | **0.9128*** | **0.9096*** | **0.9295*** |
| Douban | MLP | 0.7689 | 0.7945 | 0.7567 | 0.7732 | 0.7641 | 0.7894 |
| | DeepWalk | 0.8084 | 0.8301 | 0.7995 | 0.8054 | 0.8295 | 0.8464 |
| | node2vec | 0.8545 | 0.8623 | 0.8304 | 0.8416 | 0.8578 | 0.8594 |
| | metapath2vec | 0.8709 | 0.8901 | 0.8593 | 0.8648 | 0.8609 | 0.8783 |
| | DeepWalk+fea | 0.8535 | 0.8795 | 0.8347 | 0.8578 | 0.8548 | 0.8693 |
| | node2vec+fea | 0.8994 | 0.9045 | 0.8732 | 0.8958 | 0.8896 | 0.8935 |
| | metapath2vec+fea | 0.9248 | 0.9309 | 0.8998 | 0.9134 | 0.8975 | 0.9104 |
| | GCN | 0.9032 | 0.9098 | 0.8934 | 0.9123 | 0.9032 | 0.9112 |
| | GAT | 0.9214 | 0.9385 | 0.8987 | 0.9103 | 0.8998 | 0.9145 |
| | HAN | 0.9321 | 0.9523 | 0.9096 | 0.9221 | 0.9098 | 0.9205 |
| | TrustMF | 0.9034 | 0.9342 | 0.8798 | 0.9054 | 0.9002 | 0.9145 |
| | DiffNet | 0.9509 | 0.9634 | 0.9005 | 0.9259 | 0.9024 | 0.9301 |
| | SIAN | **0.9742*** | **0.9873*** | **0.9139*** | **0.9429*** | **0.9171*** | **0.9457*** |
| FWD | MLP | 0.5094 | 0.5182 | 0.1883 | 0.1932 | 0.2205 | 0.2302 |
| | DeepWalk | 0.5587 | 0.5636 | 0.2673 | 0.2781 | 0.1997 | 0.2056 |
| | node2vec | 0.5632 | 0.5712 | 0.2674 | 0.2715 | 0.2699 | 0.2767 |
| | metapath2vec | 0.5744 | 0.5834 | 0.2651 | 0.2724 | 0.4152 | 0.4244 |
| | DeepWalk+fea | 0.5301 | 0.5433 | 0.2689 | 0.2799 | 0.2377 | 0.2495 |
| | node2vec+fea | 0.5672 | 0.5715 | 0.2691 | 0.2744 | 0.3547 | 0.3603 |
| | metapath2vec+fea | 0.5685 | 0.5871 | 0.2511 | 0.2635 | 0.4698 | 0.4935 |
| | GCN | 0.5875 | 0.5986 | 0.2607 | 0.2789 | 0.4782 | 0.4853 |
| | GAT | 0.5944 | 0.6006 | 0.2867 | 0.2912 | 0.4812 | 0.4936 |
| | HAN | 0.5913 | 0.6025 | 0.2932 | 0.3011 | 0.4807 | 0.4937 |
| | TrustMF | 0.6001 | 0.6023 | 0.3013 | 0.3154 | 0.5298 | 0.5404 |
| | DiffNet | 0.6418 | 0.6594 | 0.3228 | 0.3379 | 0.6493 | 0.6576 |
| | SIAN | **0.6845*** | **0.6928*** | **0.3517*** | **0.3651*** | **0.6933*** | **0.7018*** |

#### 9.5.4.2 Experimental Results

We adopt three widely used metrics AUC, F1 and Accuracy to evaluate performance. The results w.r.t. the dimension of latent representation are reported in Table 9.8, from which we have the following findings.

(1) SIAN outperforms all baselines in all metrics on three datasets with statistical significance ($p < 0.01$) under paired $t$-test. It indicates that SIAN can well capture user core concerns from multifaceted factors in FER. The improvements derive from both high-quality node representations generated from node- and type-level attentive aggregations, and the social influence coupler that digs out what users are socially inclined to.

(2) Compared with the graph neural network methods, the impressive improvements of SIAN proves the effectiveness of the node- and type-level attentive aggregations. Especially, SIAN achieves better performances than HAN which is also designed for heterogeneous graphs with a two-level aggregation. It is because that the type-level attentive aggregation in SIAN captures heterogeneous information in multiple aspects, without being limited by the predefined meta-paths used in HAN. Moreover, the improvements also indicate the significance of our social influence coupler in FER.

(3) Social recommendation baselines also achieve promising performances, which further substantiates the importance of social influence in FER. Compared with baselines which only treat social relations as side information, the improvements imply that the friend referral factor may take the dominating position in FER, which should be carefully modeled. In particular, our SIAN achieves the best performance, reconfirming the capability of our social influence coupler in encoding diverse social factors for FER.

#### 9.5.4.3 Analysis on Social Influence in FER

We have verified that FRC is the most essential factor in FER. However, a friend could impact user from different aspects (e.g., authority or similarity). Next, we show how different user attributes affect user behaviors in FER.

**Evaluation Protocol.** The attention in social influence coupler reflects the importance of different friends. We assume that the friend $v$ having the highest attention value (i.e., $d_{u \leftarrow \langle v,i \rangle}$ in Eq. 9.6) is the most influential friend w.r.t. item $i$ for user $u$, and all of $v$'s attribute values are equally regarded as contributing to the influence. Given a user attribute and a user group, we define the *background distribution* by counting the attribute values of all friends in FRCs of users in this group, and also define the *influence distribution* by counting the attribute values of the most influential friends of users in the group. Thus, the background distribution represents the characteristics of general friends of this user group, while the influence distribution represents the characteristics of the most influential friends of this user group. If the two distributions perfectly agree with each other, this attribute is not a key social factor in influencing this user group. In contrast, the differences between the two

distributions imply how much this attribute is a key social factor, and how its different values affect user behaviors.

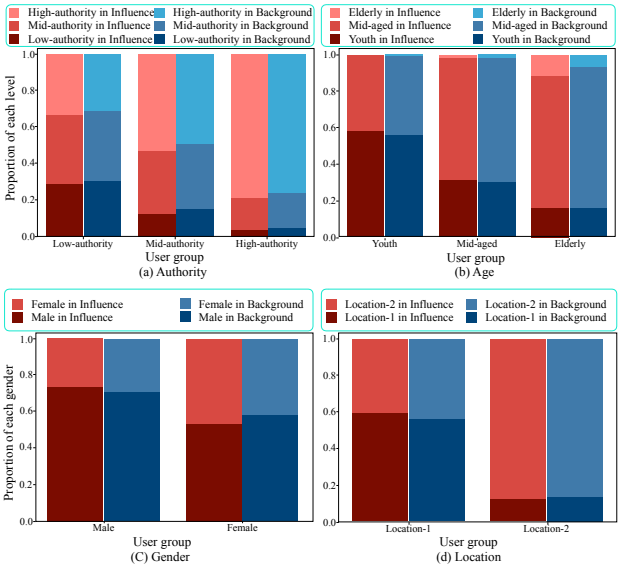



**Fig. 9.15** Social influence analysis w.r.t user attributes. For each attribute and user group (e.g., the authority and the low-authority group in (a)), the left is the influence distribution while the right is the background distribution. In each bar, the height of each different-colored segment means the proportion of an attribute value in the influence or background distribution. Best read in color.

**Results and Analysis.** As shown in Fig. 9.15, we find out the following:

(1) In Fig. 9.15a, we observe that user behaviors are more influenced by their friends who are more authoritative, regardless of what authority the user him/herself has. In all three user groups of varying authority, the proportion of high-authority in the influence distribution is larger than that in the background distribution. For instance, in the mid-authority user group, the top red block (high-authority influence) is larger than the top blue one (high-authority background), which implies that high-authority friends are more influential for mid-authority users. The result is not surprising as users are usually more susceptible and easy to be affected by authoritative persons, which is consistent with common sense.

(2) We also conduct several analyses on influences w.r.t. other user attributes. We find that *users are easy to be influenced by their friends which are similar to themselves*. Specifically, Fig. 9.15b shows that people like items recommended by their peers, especially for the youth and the elderly; meanwhile, Fig. 9.15c and d show that users tend to watch articles recommended by their friends with the same gender or location. Recommendation with user similarity, which has been widely assumed in collaborative filtering, is still classical even in FER.

The more detailed method description and experiment validation can be seen in [20].

## 9.6 Conclusions

As HG representation has a great power to fuse heterogeneous information, it has become one of the major techniques to apply HG analysis to real-world applications. This chapter presents several advanced HG representation methods applying to E-commercial systems and online social networks. Particularly, we first study the cash-out user detection problem and propose the HACUD, which is a hierarchical heterogeneous GNN method. The model could extract the user's structural features through a hierarchical attention mechanism. In addition, we study a newly emerged problem in E-commercial system, named intent recommendation, and propose a novel metapath-guided heterogeneous GNN method, called MEIRec. The MEIRec model learns users' and queries' embeddings through multiple predefined meta-paths. Furthermore, we study the share recommendation problem, which is a unique recommendation paradigm in social e-commerce, and propose a tripartite heterogeneous GNN, named HGSRec. Different from MEIRec model concatenating multiple learned embeddings, the proposed method aggregates multiple embeddings through a co-attention mechanism. Apart from E-commerce, we also study the friend-Enhanced recommendation problem in online social networks and propose a novel social influence attentive neural network method. The experiments of these methods solidly validate the effectiveness of HG embedding methods on real-world applications.

More interesting future applications are worth being exploited on HG representation. For example, in the biological area, there are multiple relations between gene expression and phenotype, which can be naturally constructed as an HG. Besides, in the software engineering area, there are complex relations among test sample, requisition form, and problem form, which can be naturally modeled as an HG. Therefore, HG representation is expected to open up broad prospects for these new application areas and become a promising analytical tool.

## References

1. Antoine, B., Nicolas, U., Alberto, G.D., Jason, W., Oksana, Y.: Translating embeddings for modeling multi-relational data. In: NeurIPS, pp. 2787–2795 (2013)
2. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. ICLR (2015)
3. Bergstra, J.S., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. In: NeurIPS, pp. 2546–2554 (2011)
4. Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system. In: KDD, pp. 785–794 (2016)

5. Dai, H., Dai, B., Song, L.: Discriminative embeddings of latent variable models for structured data. In: ICML, pp. 2702–2711 (2016)
6. Dong, Y., Chawla, N.V., Swami, A.: metapath2vec: Scalable representation learning for heterogeneous networks. In: KDD, pp. 135–144 (2017)
7. Dong, Y., Chawla, N.V., Swami, A.: metapath2vec: Scalable representation learning for heterogeneous networks. In: KDD (2017)
8. Fan, S., Zhu, J., Han, X., Shi, C., Hu, L., Ma, B., Li, Y.: Metapath-guided heterogeneous graph neural network for intent recommendation. In: KDD, pp. 2478–2486 (2019)
9. Friedman, J.H.: Greedy function approximation: a gradient boosting machine. Annals of statistics pp. 1189–1232 (2001)
10. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: AISTATS, pp. 249–256 (2010)
11. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: KDD, pp. 855–864 (2016)
12. Hu, B., Zhang, Z., Shi, C., Zhou, J., Li, X., Qi, Y.: Cash-out user detection based on attributed heterogeneous information network with a hierarchical attention mechanism. In: AAAI, pp. 946–953 (2019)
13. Ji, H., Zhu, J., Wang, X., Shi, C., Wang, B., Tan, X., Li, Y., He, S.: Who you would like to share with? a study of share recommendation in social e-commerce. In: AAAI (2021)
14. Jolly, A.: Lemur social behavior and primate intelligence. Science pp. 501–506 (1966)
15. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. In: ICLR (2015)
16. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR (2017)
17. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR (2017)
18. Liang, J., Jacobs, P., Sun, J., Parthasarathy, S.: Semi-supervised embedding in attributed networks with outliers. In: SDM, pp. 153–161 (2018)
19. Lobo, J.M., Jiménez-Valverde, A., Real, R.: Auc: a misleading measure of the performance of predictive distribution models. Global ecology and Biogeography **17**(2), 145–151 (2008)
20. Lu, Y., Xie, R., Shi, C., Fang, Y., Wang, W., Zhang, X., Lin, L.: Social influence attentive neural network for friend-enhanced recommendation. In: ECML-PKDD (2020)
21. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: KDD, pp. 701–710 (2014)
22. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: KDD, pp. 701–710 (2014)
23. Qu, M., Tang, J., Shang, J., Ren, X., Zhang, M., Han, J.: An attention-based collaboration framework for multi-view network representation learning. In: CIKM, pp. 1767–1776 (2017)
24. Shi, C., Hu, B., Zhao, W.X., Philip, S.Y.: Heterogeneous information network embedding for recommendation. TKDE **31**(2), 357–370 (2019)
25. Shi, C., Li, Y., Zhang, J., Sun, Y., Philip, S.Y.: A survey of heterogeneous information network analysis. IEEE Transactions on Knowledge and Data Engineering **29**(1), 17–37 (2017)
26. Sun, Y., Han, J., Yan, X., Yu, P.S., Wu, T.: Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. In: VLDB, pp. 992–1003 (2011)
27. Tieleman, T., Hinton, G.: Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural networks for machine learning **4**(2), 26–31 (2012)
28. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: ICLR (2018)
29. Wang, X., He, X., Wang, M., Feng, F., Chua, T.S.: Neural graph collaborative filtering. In: SIGIR, pp. 165–174 (2019)
30. Wang, X., Ji, H., Shi, C., Wang, B., Ye, Y., Cui, P., Yu, P.S.: Heterogeneous graph attention network. In: WWW, pp. 2022–2032 (2019)
31. Wang, X., Ji, H., Shi, C., Wang, B., Ye, Y., Cui, P., Yu, P.S.: Heterogeneous graph attention network. In: WWW, pp. 2022–2032 (2019)

32. Wang, Z., Liao, J., Cao, Q., Qi, H., Wang, Z.: Friendbook: a semantic-based friend recommendation system for social networks. IEEE transactions on mobile computing **14**(3), 538–551 (2014)
33. Wu, L., Sun, P., Fu, Y., Hong, R., Wang, X., Wang, M.: A neural influence diffusion model for social recommendation. In: SIGIR, pp. 235–244 (2019)
34. Xiao, W., Houye, J., Chuan, S., Bai, W., Peng, C., P., Y., Yanfang, Y.: Heterogeneous graph attention network. In: WWW (2019)
35. Yang, B., Lei, Y., Liu, J., Li, W.: Social collaborative filtering by trust. IEEE transactions on pattern analysis and machine intelligence pp. 1633–1647 (2016)
36. Zhang, Z., Yang, H., Bu, J., Zhou, S., Yu, P., Zhang, J., Ester, M., Wang, C.: Anrl: Attributed network representation learning via deep neural networks. In: IJCAI, pp. 3155–3161 (2018)