Learning Meta-path-aware Embeddings for Recommender Systems

Qianxiu Hao[1,2], Qianqian Xu[1*], Zhiyong Yang[2], Qingming Huang[1,2,3,4*]
[1]Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, CAS, Beijing, China
[2]School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing, China
[3]Key Laboratory of Big Data Mining and Knowledge Management, Chinese Academy of Sciences, Beijing, China
[4]Artificial Intelligence Research Center, Peng Cheng Laboratory, Shenzhen, China
qianxiu.hao@vipl.ict.ac.cn,xuqianqian@ict.ac.cn,yangzhiyong@iie.ac.cn,qmhuang@ucas.ac.cn

ABSTRACT

Heterogeneous information networks (HINs) have become a popular tool to capture complicated user-item relationships in recommendation problems in recent years. As a typical instantiation of HINs, meta-path is introduced in search of higher-level representations of user-item interactions. Though remarkable success has been achieved along this direction, existing meta-path-based recommendation methods face at least one of the following issues: 1) existing methods merely adopt simple meta-path fusion rules, which might be insufficient to exclude inconsistent information of different meta-paths that may hurt model performance; 2) the representative power is limited by shallow/stage-wise formulations. To solve these issues, we propose an end-to-end and unified embedding-based recommendation framework with graph-based learning. To address 1), we propose a flexible fusion module to integrate meta-path-based similarities into relative similarities between users and items. To address 2), we take advantage of the powerful representative ability of deep neural networks to learn more complicated and flexible latent embeddings. Finally, empirical studies on real-world datasets demonstrate the effectiveness of our proposed method.

CCS CONCEPTS

• **Information systems** → **Social recommendation**; **Recommender systems**;

KEYWORDS

Meta Path, Heterogeneous Information Network, Recommendation System

ACM Reference Format:
Qianxiu Hao[1,2], Qianqian Xu[1*], Zhiyong Yang[2], Qingming Huang[1,2,3,4*]. 2021. Learning Meta-path-aware Embeddings for Recommender Systems. In *Proceedings of the 29th ACM International Conference on Multimedia (MM '21), October 20–24, 2021, Virtual Event, China.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3474085.3475407

MM '21, October 20–24, 2021, Virtual Event, China

ACM ISBN 978-1-4503-8651-7/21/10…$15.00
https://doi.org/10.1145/3474085.3475407

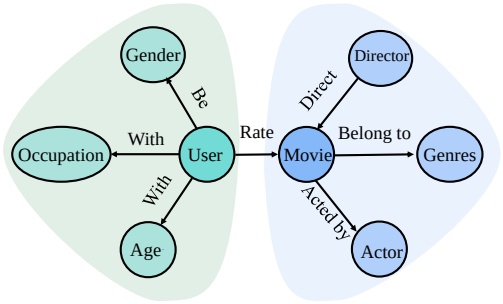


Figure 1: An example of an HIN schema built on MovieLens.

1 INTRODUCTION

Nowadays, Recommender System (RS) has emerged as a popular method which has brought promising benefits to a wide spectrum of applications. Though related studies have made great achievements, the majority of them focus on user-item feedbacks and simple content information, which unavoidably faces issues such as data sparsity [9, 31] and cold start [1, 13]. Recent years, researchers start to utilize extra supervisions in the form of user and/or item relationships, known as heterogeneous information networks (HINs) [10, 12, 20, 22], to alleviate some of the above issues. Generally speaking, an HIN refers to an information network that consists of different kinds of interconnected entities, examples including bibliographic information networks, social media networks, etc. Along with this new wave, most studies explore HINs with the help of meta-paths [3, 17, 19, 25, 32], which are known for their flexibility and strong representative ability. Specifically, a meta-path is a path in a graph where each node owns a type attribute and each edge owns a relation attribute. In the following, we provide an intuitive example showing that meta-paths embrace much more complicated information than user-item feedback data.

Example. *In the schema shown in Fig.1, each node represents a type of objects (age, occupation, directors and actors), and each edge represents a specific relationship (with, direct, acted by). Based on the schema, we could derive a variety of meta-paths starting with a user/item node, passing through several objects, and then terminate at an item/user node. To name a few,* user $\xrightarrow{rate}$ movie, user $\xrightarrow{rate}$ movie $\xrightarrow{be\ directed\ by}$ director $\xrightarrow{direct}$ movie, *etc. For each meta-path, it depicts a higher-order relationship involving its user-item terminals. Since user-item feedback is actually a length-2 special case of meta-path,*

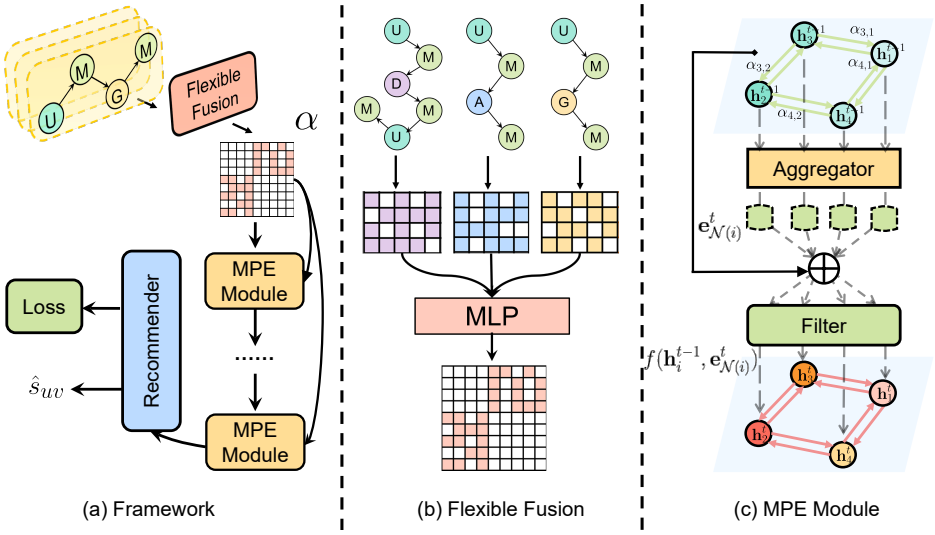


Figure 2: (a) The overall framework. (b) Flexible Fusion module converts multiple meta-paths into a relative similarity graph α. In the figure, U is short for a user object, and the other abbreviations represent: M for movies, D for directors, A for actors, and G for genres. (c) For each time step t, in MPE module, an aggregator first assembles message from neighbors $\mathcal{N}(i)$ of a node i, producing a neighborhood embedding $e^t_{\mathcal{N}(i)}$, under the guidance of α. Then a filter takes inputs as both current node embedding h_i^{t-1} and $e^t_{\mathcal{N}(i)}$ to conduct embedding updating. After several time steps, the final embeddings are inputted into the recommendation module for predicting rating $\hat{s}_{uv}$.

this implies that introducing a more general form of meta-path would help us dive deeper into the complex world of user-item interactions.

There are often multiple meta-paths co-existing in a complicated dataset with different meta-paths indicating different semantics. For example, $user \xrightarrow{rate} movie \xrightarrow{be\ directed\ by} director \xrightarrow{direct} movie$ indicates the situation that a user explores new movies by directors, while $user \xrightarrow{be\ friend\ of} user \xrightarrow{rate} movie$ indicates that a user explores new movies by his/her friends. How to effectively capture and assemble multiple complicated semantics for recommendation is a challenging problem. There have been some efforts to tackle the problem. Early work [14, 33] utilized meta-path-based similarities to perform regularization on user/item representations, so that more similar user-user/item-item pairs are enforced to be closer in the latent space. Later, some work first learned user/item representations along each path and then fused them [17, 29, 32], or conducted weighted summary of different meta-path-based similarities [19], to predict rating scores. Though having witnessed great successes, they only exploited stage-wise or shallow models, suffering from a limited representation ability. Recently, there is a surge of efforts that first apply meta-path-based random walks to sample heterogeneous neighbors and then adopt graph representation learning methods to learn multi-views of embeddings and then fuse them together. However, since the fusion process takes place in the late stage of the whole training, this paradigm also suffers from the sparity

Seeing the aforementioned issues, our motivation is to propose a unified end-to-end framework that jointly conducts flexible fusion scheme learning, meta-path-aware user/item representation learning, and effective rating prediction. The main contributions of this paper are three-fold:

- To effectively assemble meta-path-specific semantics, we design a Flexible Fusion Module that could learn arbitrary fusing strategies driven by downstream tasks.
- To incorporate meta-path semantics for recommendations, a Meta-Path-aware based Embedding (MPE) Module is then proposed to utilize the fused meta-path semantics to guide the user/item information propagation.
- We also propose a sampling strategy to enable mini-batch gradient descent to fit large-scale datasets. Empirical studies on real datasets demonstrate the effectiveness of the proposed method.

There is a parallel direction closely related to our studies, i.e., the knowledge graph (KG) based models [2, 15, 23, 30]. Since KGs may involve many types of entities or relations that are not helpful for recommendation, conventional KG-based embedding methods may contain a large amount of redundant information in the learned embeddings, which is not efficient. Our method, however, integrates human knowledge of meta-paths which is specifically designed for recommendation, are thus more efficient.

2 RELATED WORK

Recently, some researchers resort to HINs to enhance recommending quality with the help of meta-paths. Roughly speaking, meta-path-based recommendation models fall into two branches, i.e., matrix factorization (MF) based and network-embedding-based methods. Among the matrix factorization methods, the earliest work [28] proposed regularization terms to incorporate meta-paths

that terminate with users in an MF framework. Later, [18] and [33] followed the idea and further proposed dual regularizations to incorporate meta-paths that terminate with two users and that with two items. Other studies treat path-based similarity matrices equivalently as rating matrix, and directly assemble them to approximate the rating matrix [19], or first factorized them to obtain groups of low dimensional representations and then integrate them to approximate rating matrix [29, 32]. Recently, [17] proposed meta-path-based random walks to generate homogeneous neighbors, based on which meta-path-specific representations are learned. Then they designed fusion functions to integrate meta-path-specific representations and incorporated them into an extended MF predictor.

Apart from MF-based methods, other techniques are also employed to embrace HIN-based recommendations. [8] and [26] both developed attention mechanisms to leverage different semantics of meta-paths. The difference is that [26] adopted the attention mechanism to learn the preferences of users on different meta-paths. While [8] employed an attention mechanism to learn explicit representations of meta-paths, which are concatenated with users' representations and items' representations for preference learning. [7] proposed an attention mechanism to model the most informative embeddings for users and items. Then they learned effective relation representations to capture rich information in HIN.

As an important distinction from these methods, our method is able to learn flexible meta-path fusing schemes to explore useful semantics from multiple paths, rather than assign explicit weights to meta-paths for path aggregation. Besides, in our model, one could flexibly initialize the node features according to nodes' content information/attributes, while the existing HIN-based recommendation algorithms often fail to incorporate such node semantics.

3 PRELIMINARY

Before entering into the discussion of methodology, let us review some related concepts about the meta-path in HINs.

Definition 1 (Network Schema [21]). *Let $\mathcal{A}$ and $\mathcal{R}$ be the sets of node types and link types, respectively. The network schema $\Pi = (\mathcal{A}, \mathcal{R})$ is a meta template for an HIN $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with an object type mapping $\phi : \mathcal{V} \to \mathcal{A}$ and a link type mapping $\psi : \mathcal{E} \to \mathcal{R}$, s.t., $|\mathcal{A}|+|\mathcal{R}| > 2$ (see Fig. 1).*

Definition 2 (Meta-path [21]). *A meta-path $\mathcal{P}$ is defined on the network schema $\Pi = (\mathcal{A}, \mathcal{R})$ of an HIN, and is a semantic pattern in the form of $A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \cdots \xrightarrow{R_l} A_{l+1}$. A meta-path describes a composite relation $R = R_1 \circ R_2 \circ \cdots \circ R_l$ between two terminal objects with type A_1 and A_{l+1}, where $\circ$ denotes the composition operator.*

Since meta-paths are semantic patterns defined on network schema, there may be many different entity sequences (called *meta-path instances*) following a same meta-path pattern. More concretely, *John* $\xrightarrow{befriend\ of}$ *Henry* $\xrightarrow{rate}$ *The Avengers* is a meta-path instance following the meta-path *user* $\xrightarrow{befriend\ of}$ *user* $\xrightarrow{rate}$ *movie*.

Intuitively, each meta-path indicates a specific kind of semantics. How to capture such semantics is a challenging problem. In the literature, semantics of such kind are often formulated as meta-path-based similarities. In this paper, we follow the well-known PathCount [21] and adopted its normalized version. PathCount similarities are calculated through a commuting matrix, which is defined as:

Definition 3 (Commuting Matrix [21]). *Given an HIN $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and its network schema $\Pi = (\mathcal{A}, \mathcal{R})$, and denote the set of nodes of type A_i as $\mathcal{V}_{A_i}$, denote $\boldsymbol{W}^{ij} \in \{0,1\}^{|\mathcal{V}_{A_i}|\times|\mathcal{V}_{A_j}|}$ as the adjacency matrix between nodes of type A_i and nodes of type A_j, where entry $W^{ij}_{mn} = 1$ means there is a direct link between node m (of type A_i) and node n (of type A_j), $W^{ij}_{mn} = 0$ otherwise. A commuting matrix $\boldsymbol{M}$ of a meta-path $\mathcal{P} = (A_1 \xrightarrow{R_1} A_2 \cdots \xrightarrow{R_l} A_{l+1})$ is defined as $\boldsymbol{M} = \boldsymbol{W}^{12}\boldsymbol{W}^{23}\cdots\boldsymbol{W}^{l(l+1)}$.*

PathCount similarity between i-th entity (of type A_1) and j-th entity (of type A_{l+1}) is then defined as M_{ij}, which is the entry along i-th row and j-th column of $\boldsymbol{M}$. Clearly, M_{ij} represents the number of path instances between i-th and j-th entity following meta-path $\mathcal{P} = (A_1 \xrightarrow{R_1} A_2 \cdots A_{l+1})$. We adopt its normalized version, i.e. $\frac{M_{ij}}{\sqrt{\sum_k M_{ik}}\sqrt{\sum_k M_{kj}}}$ and denote it as $\hat{M}_{ij}$. It can be seen that the adopted normalized PathCount similarity $\hat{M}_{ij}$ can be seen as the accessibility or similarity for entity j w.r.t entity i. When $\hat{M}_{ij}$ is large, we could expect that entity i has a large possibility to access entity j, which motivates us to incorporate it in the representation learning process to enlarge the possibility of a user to interact an item. In the following section, we will elaborate on how to incorporate such accessibility or similarity into the representation learning of user/item embeddings.

4 METHODOLOGY

Let $\mathcal{U}$ and $\mathcal{I}$ be the set of all users and items, respectively, and $\mathcal{S}$ be the set of all rating scores. Moreover, define the corresponding cardinality $m = |\mathcal{U}|$, $n = |\mathcal{I}|$. In this paper, our goal is to solve the following problems:

Problem. *Given the rating set $\mathcal{Y} = \{(u, v, s)|u \in \mathcal{U}, v \in \mathcal{I}, s \in \mathcal{S}\}$, an HIN $\mathcal{G}$, and the related meta-path set $\{\mathcal{P}_1, \cdots, \mathcal{P}_L\}$, where L is the number of meta-paths, can we find a unified model to **1)** integrate meta-path semantics into a unified user-item similarity representation, **2)** effectively refine meta-path-aware user/item embeddings, and **3)** predict unknown ratings for the recommender system?*

In this section, we propose a unified meta-path-aware embedding-based framework as a solution. Generally speaking, our framework includes the following three modules, as shown in Figure 2, to answer the above three questions:

(M1) *Flexible Fusion Module.* We develop the flexible fusion module to learn a proper assembling strategy. M1 is able to integrate diversified meta-path-based similarities into a unified user-item relative similarity graph.

(M2) *Meta-path Embedding Module.* We propose the meta-path-aware embedding module (MPE) to incorporate the learned user-item relative similarities into user/item embeddings. MPE provides a recursive rule to learn meta-path-aware user/item representations based on the similarity graph attained from M1.

(M3) *Prediction Module.* Finally, we provide a prediction module to obtain the score as an estimation for ratings.

4.1 Flexible Fusion Module

We begin with the flexible fusion module to integrate diversified semantics coming from multiple meta-paths into a unified *relative similarity graph*. Suppose there are L meta-paths $\{\mathcal{P}_1, \mathcal{P}_2, \cdots, \mathcal{P}_L\}$, we can calculate the corresponding normalized PathCount matrices as $\{\hat{\boldsymbol{M}}^{(1)}, \hat{\boldsymbol{M}}^{(2)}, \cdots, \hat{\boldsymbol{M}}^{(L)}\}$. Then for each user-item pair (u, v), we define a vector $\boldsymbol{r}_{uv}$, which concatenates different path-based similarities:

$$\boldsymbol{r}_{uv} = \left[\hat{M}^{(1)}_{uv}, \hat{M}^{(2)}_{uv}, \cdots, \hat{M}^{(L)}_{uv} \right]. \tag{1}$$

Herein, $\hat{M}^{(l)}_{uv}$ is the entry along u-th row and v-th column of the similarity matrix $\hat{\boldsymbol{M}}^{(l)}$. In this paper, we view $\boldsymbol{r}_{uv}$ as a vector representing the complicated relationship between u and v. Next, we extract a unified similarity representation from $\boldsymbol{r}_{uv}$ in terms of a unified relative similarity between u and v. To do this, we first build an auxiliary bipartite graph, where we introduce $\{\boldsymbol{r}_{uv}\}_{u,v}$ into the edge representation of the graph. The graph is denoted as $\mathcal{BG}(\mathcal{U}, \mathcal{I}, \mathcal{E}, \mathcal{R})$, where its edge representation $\mathcal{R} = \{\boldsymbol{r}_{uv} | u \in \mathcal{U}, v \in \mathcal{I}\}$, and edge set $\mathcal{E} = \{(u, v), (v, u) | u \in \mathcal{U}, v \in \mathcal{I}, \text{if } \boldsymbol{r}_{uv} \neq \boldsymbol{0}\}$. On top of $\mathcal{BG}$, we can also define the *neighborhood* $\mathcal{N}(i)$ to be $\{j | (i, j) \in \mathcal{E}\}$, for any given node $i \in \mathcal{U} \cup \mathcal{I}$.

To induce the relative similarities, let us take a closer look at the entries of vector $\boldsymbol{r}_{uv}$. For each path $\mathcal{P}_l$, $M^{(l)}_{uv}$ is positively-related to the number of path instances between u and v. In this way, $M^{(l)}_{uv}$ could be regarded as a proper metric for similarity. Hence, we consider the final relative similarity as a function of $\left\{M^{(l)}_{uv}\right\}_{l=1}^{L}$. More precisely, let α_{uv} denote the relative similarity of v and u, we model $\alpha_{uv} = f(\boldsymbol{r}_{uv})$. Due to the well-known universal approximation property of neural networks [6], we adopt multiple layer perception (MLP) to learn a unified similarity. More specifically, given edge representation $\boldsymbol{r}_{uv}$ as inputs, MLP outputs an initial representation e_{uv}, where

$$e_{uv} = \quad \text{MLP}(\boldsymbol{r}_{uv}), \quad \forall (u, v) \in \mathcal{E}. \tag{2}$$

Although e_{uv} represents a comprehensive relationship between nodes, it can not be directly used to capture asymmetric relative similarities. In fact, the relative similarity of v to u should not be equal to that of u to v, since u and v do not share the same neighborhood. To preserve the asymmetry, we adopt a normalization strategy within neighborhoods, which yields direction-specific importances α_{uv} and α_{vu}:

$$\begin{aligned} \alpha_{uv} &= \text{softmax}_v(e_{uv}) = \frac{\exp(e_{uv})}{\sum_{j \in \mathcal{N}(u)} \exp(e_{uj})}, \\ \alpha_{vu} &= \text{softmax}_u(e_{uv}) = \frac{\exp(e_{uv})}{\sum_{i \in \mathcal{N}(v)} \exp(e_{iv})}. \end{aligned} \tag{3}$$

Putting together all α_{uv} and α_{vu}, we get a final bipartite similarity graph[1] $\boldsymbol{\alpha} = \begin{bmatrix} \boldsymbol{0}, & \boldsymbol{\alpha}_{u \to v} \\ \boldsymbol{\alpha}_{v \to u}, & \boldsymbol{0} \end{bmatrix}$, where $\boldsymbol{\alpha}_{u \to v}$ is a matrix $\in \mathbb{R}^{m \times n}$ containing all the user-to-item importance α_{uv}s, while $\boldsymbol{\alpha}_{u \to v} \in \mathbb{R}^{n \times m}$ contains α_{vu}s.

In this module, we end up with a relative similarity graph defined as above, which is fed to the next module to learn meta-path-aware embeddings.

[1] Actually, $\boldsymbol{\alpha}$ is the adjacency matrix of the graph. But in this paper, we refer to the relative similarity graph as $\boldsymbol{\alpha}$ for simplicity.

4.2 Meta-path-aware Embedding Module

Suppose the raw features of users are $\boldsymbol{X}^U \in \mathbb{R}^{m \times p}$, and those of items are $\boldsymbol{X}^I \in \mathbb{R}^{n \times q}$, where p, q are the corresponding feature dimensions, respectively. Since user/item raw features are located in heterogeneous spaces, their feature dimension p and q may not be equal. So we first transform raw features into a common latent space to make sure the initial user/item embeddings $\boldsymbol{H}^0$ are of the same dimension. The transformation is conducted as:

$$\boldsymbol{H}^0 = \begin{bmatrix} \boldsymbol{X}^U \boldsymbol{W}^U \\ \boldsymbol{X}^I \boldsymbol{W}^I \end{bmatrix}, \tag{4}$$

where $\boldsymbol{W}^U \in \mathbb{R}^{p \times k}$ and $\boldsymbol{W}^I \in \mathbb{R}^{q \times k}$ are the projectors to be learned, with $k \ll p, q$ being the dimensionality of the latent space. $\boldsymbol{H} \in \mathbb{R}^{(m+n) \times k}$ is the latent feature matrix consisting of user/item transformed features. In the rest of this subsection, we will treat user and item nodes equivalently without distinguishing them.

On top of the common latent space, the module produces the final embeddings which could preserve both input feature information and topology information. Inspired by [5], this is implemented by an information propagation mechanism over the *relative similarity graph* $\boldsymbol{\alpha}$. To leverage comprehensive information propagation across nodes, we construct a message-passing procedure in a recursive way. At each time $t = 1, \cdots, \mathcal{T}$, we introduce a two-stage propagation mechanism. For any fixed time t, a new neighborhood embedding $\boldsymbol{e}^t_{\mathcal{N}(i)}$ is generated by aggregating the latest embeddings at $t-1$ from the nodes in the neighborhood of node i. This is expressed as Eq.(5). In the second stage, we perform a filtering operation based on $\boldsymbol{h}^{t-1}_i$ and $\boldsymbol{e}^t_{\mathcal{N}(i)}$ to pick up useful signals and form its new embedding. This is expressed as Eq.(6):

$$\boldsymbol{e}^t_{\mathcal{N}(i)} = \text{Aggregator}\left(\left\{\boldsymbol{h}^{t-1}_j, \forall j \in \mathcal{N}(i)\right\}, \boldsymbol{\alpha}\right), \tag{5}$$

$$\boldsymbol{h}^t_i = \text{Filter}(\boldsymbol{h}^{t-1}_i, \boldsymbol{e}^t_{\mathcal{N}(i)}) = \sigma\left(\boldsymbol{W}^t \left[\boldsymbol{h}^{t-1}_i || \boldsymbol{e}^t_{\mathcal{N}(i)}\right]\right) \tag{6}$$

Herein, $||$ represents concatenation operation, and σ is an activation function for the filter. As is shown in the above equations, at a given step t, the embedding generation process has two phases. First, an aggregator Eq.(5) assembles messages coming from the neighbors, producing a single vector $\boldsymbol{e}^t_{\mathcal{N}(i)}$. Then the filter performs a gate operation to extract useful information that might help in the downstream tasks. As the process advances, user/item embeddings gain more and more information from their higher-order neighbors in the graph. With the embedding process provided, we now give our instantiation of the aggregator adopted in this paper, which takes a form in the following:

$$\text{Aggregator}\left(\left\{\boldsymbol{h}^{t-1}_j, \forall j \in \mathcal{N}(i)\right\}, \boldsymbol{\alpha}\right) = \sum_{j \in \mathcal{N}(i)} \alpha_{ij} \boldsymbol{h}^{t-1}_j. \tag{7}$$

So far, we have gained user/item embeddings from meta-path semantics. The MPE module then feeds the resulting embeddings to the next module to produce the final scores.

Discussion. Eq.5 and Eq. 6 work in a message-passing way, similar to many GNN-based methods. But the important difference is that during the message-passing, we model the aggregating factors by the $\boldsymbol{\alpha}$, which reflects the unified similarities between users and items. Whereas, the aggregating factors of traditional GNN-based methods do not contain such meta-path semantics to enhance the

message-passing. From Eq.5 and Eq. 6, we can see that the distance between the final embeddings of user u and item v that have large α_{uv} is shortened. If the interaction between user u and item v that has large similarity is negative, the optimizer will automatically pay more attention to such negative pairs, and depart them farther away.

4.3 Prediction Module

Having finished modeling the embeddings of users and items guided with meta-paths, we are ready to establish our prediction module. Given user/item embeddings $\boldsymbol{H}^{\mathcal{T}}$ at time step $\mathcal{T}$, we employ a fully connected layer to construct a final low dimensional embedding which could be represented as:

$$\boldsymbol{z}_i = \sigma(\boldsymbol{V}\boldsymbol{h}_i^{\mathcal{T}} + \boldsymbol{b}), \forall i \in \mathcal{U} \cup \mathcal{I}, \tag{8}$$

where $\boldsymbol{h}_i^{\mathcal{T}}$ is the final step embedding for node i. $\boldsymbol{V} \in \mathbb{R}^{d \times k}$ and $\boldsymbol{b} \in \mathbb{R}^d$ are learnable parameters, and $d < k$ is the dimension of final representations. Finally, to predict the ratings, we adopt a scoring function $s(z_u, z_v)$ to learn a proper score $\hat{s}_{uv}$, which estimates the true rating. In this paper, we propose two specific realizations of the function $s(.,.)$:

Linear Model. In this model, we model the rating from user u to item v as:

$$\hat{s}_{uv} = \boldsymbol{a}^\top [\boldsymbol{z}_u || \boldsymbol{z}_v], \tag{9}$$

where $\boldsymbol{a} \in \mathbb{R}^{2d}$ is the learnable weights. This model is simply the weighted sum of the embedding of a user-item pair.

Bilinear Model. In this model, we adaptively learn a flexible distance metric between user and item embeddings. The distance metric is parameterized by a learnable bilinear weight matrix $\boldsymbol{Q} \in \mathcal{R}^{d \times d}$, formulated as:

$$\hat{s}_{uv} = \boldsymbol{z}_u^\top \boldsymbol{Q} \boldsymbol{z}_v, \tag{10}$$

Compared to the linear model, the bilinear model can explore second-order interactions across user and item embeddings. Note that when $\boldsymbol{Q}$ is an identity matrix, Eq.(10) degenerates into inner products of user/item embeddings. Hence, the bilinear model is more flexible and powerful to model interactions between user and item embeddings.

Loss Function. Since our goal is to predict the rating score, we adopt the least square loss to encourage a better regression to the ground truth ratings $s^\star_{uv}$:

$$\mathcal{L} = \frac{1}{2|\mathcal{Y}_\mathcal{S}|} \sum_{(u,v,s^\star_{uv}) \in \mathcal{Y}_\mathcal{S}} (s^\star_{uv} - \hat{s}_{uv})^2, \tag{11}$$

where $\mathcal{Y}_\mathcal{S}$ is rating triplets $\{(u, v, s^\star_{uv})\}$ in a training batch.

4.4 Optimization

At first glance, we could simply resort to stochastic gradient descent to optimize our model. However, a naive sampling strategy on nodes will ruin the topological structure of the graph, leading to disastrous performance degradation. Inspired by recent studies [4, 5], we propose a node sampling strategy to preserve the topological structure. The high-level idea is that it is not a must to use the whole graph in every batch. In fact, since the connected user/items along some meta-paths may not have true interactions. Thus it is okay to only input parts of the relative similarity graph. Besides, we input all the neighboring nodes reflected by meta-path **UI** (user $\xrightarrow{rate}$ item) since it indicates true interactions. Motivated by this, for each user/item node in a mini-batch, we first add all its neighbors captured by the meta-path **UI**. Then we sample a fixed number of nodes from $\mathcal{N}(i), \forall\ i \in \mathcal{S}$ into the sampled mini-batch. The sampling strategy is depicted in Algorithm 1. We then perform a standard SGD based on the sampled mini-batch.

In practice, we adopt a vectorized implementation of the equations to take advantage of the efficient sparse matrix operation. For example, Eq.(5), (6) and (8) can be rewritten as:

$$\begin{aligned} \boldsymbol{E}^t_\mathcal{N} &= \boldsymbol{\alpha}\boldsymbol{H}^{t-1}, \\ \boldsymbol{H}^t &= \sigma(\boldsymbol{W}^t \left[\boldsymbol{H}^{t-1} || \boldsymbol{E}^{t-1}_\mathcal{N}\right]), \\ \boldsymbol{Z} &= \sigma(\boldsymbol{H}^{\mathcal{T}} \boldsymbol{V}^\top + \boldsymbol{b}) \end{aligned} \tag{12}$$

To end up this subsection, we summarize the whole learning process of our model in Algorithm 2.

Algorithm 1 Sampling Strategy

Input: User set $\mathcal{U}$, item set $\mathcal{I}$, rating triplets $\mathcal{Y}$.
Output: User subset $\tilde{\mathcal{U}}$, item subset $\tilde{\mathcal{I}}$, sampled ratings $\mathcal{Y}_\mathcal{S}$.

Sample $\mathcal{Y}_\mathcal{S}$ from $\mathcal{Y}$, get $\tilde{\mathcal{U}}_0 = \{u|(u,v,s) \in \mathcal{Y}_\mathcal{S}\}$ and $\tilde{\mathcal{I}}_0 = \{v|(u,v,s) \in \mathcal{Y}_\mathcal{S}\}$.
Initialize $\tilde{\mathcal{I}} \leftarrow \tilde{\mathcal{I}}_0$ and $\tilde{\mathcal{U}} \leftarrow \tilde{\mathcal{U}}_0$.
for each node $u \in \tilde{\mathcal{U}}_0$ **do**
  $\tilde{\mathcal{I}} \leftarrow \tilde{\mathcal{I}} \cup \{v|(u,v,s) \in \mathcal{Y}\}$
  Sample $\tilde{\mathcal{I}}_u$ from $\mathcal{N}(u)$
  $\tilde{\mathcal{I}} \leftarrow \tilde{\mathcal{I}} \cup \tilde{\mathcal{I}}_u$
end for
for each node $v \in \tilde{\mathcal{I}}_0$ **do**
  $\tilde{\mathcal{U}} \leftarrow \tilde{\mathcal{U}} \cup \{u|(u,v,s) \in \mathcal{Y}\}$
  Sample $\tilde{\mathcal{U}}_v$ from $\mathcal{N}(v)$
  $\tilde{\mathcal{U}} \leftarrow \tilde{\mathcal{U}} \cup \tilde{\mathcal{U}}_v$
end for
return $\mathcal{Y}_\mathcal{S}, \tilde{\mathcal{U}}, \tilde{\mathcal{I}}$

Algorithm 2 Training Procedure

Input: HIN $\mathcal{G}$ (contains user set $\mathcal{U}$ and item set $\mathcal{I}$), meta-path set $\mathcal{P}$, ratings $\mathcal{Y}$, raw features $\boldsymbol{X}^U$ and $\boldsymbol{X}^I$.
Output: Model parameters $\boldsymbol{\Theta}$.

Calculate normalized PathCount similarities $\{\hat{\boldsymbol{M}}^{(1)}, \cdots, \hat{\boldsymbol{M}}^{(L)}\}$ based on meta path set $\{\mathcal{P}_1, \cdots, \mathcal{P}_L\}$.
Initialize model parameters $\boldsymbol{\Theta}$
while not meet stopping criterion **do**
  Obtain mini-batch $\tilde{\mathcal{U}}, \tilde{\mathcal{I}}, \mathcal{Y}_\mathcal{S}$ by Algorithm 1
  Calculate the relative similarity graph $\boldsymbol{\alpha}$.
  Calculate initial embedding $\boldsymbol{H}^0$ and update embeddings $\boldsymbol{H}^t, t = 1, 2, ...\mathcal{T}$, according to Eq.(4) and Eq.(10), respectively.
  Predict ratings $\hat{\mathcal{Y}}_\mathcal{S}$ and calculate loss $\mathcal{L}_{\mathcal{Y}_\mathcal{S}}$ by (M3).
  Update model $\boldsymbol{\Theta} \leftarrow \boldsymbol{\Theta} - \nabla_{\boldsymbol{\Theta}} \mathcal{L}_{\mathcal{Y}_\mathcal{S}}$
end while
return $\boldsymbol{\Theta}$

Table 1: Statistics of the datasets.

Datasets	MovieLens-1M	Yelp	Amazon
#Users	6,040	5,413	7,638
#Items	3,952	4,761	4,388
#Ratings	800,167	17,802	36,300
Density	3.35%	0.07%	0.11%

Table 2: Performance Comparisons on MovieLens-1M. ↓ means that the lower value, the better. The orange color highlights the best results and the blue ones is the best results among baselines.

Training Ratio	40%		60%		80%	
Metrics	RMSE ↓	MAE ↓	RMSE ↓	MAE ↓	RMSE ↓	MAE ↓
FM	0.9721	0.7820	0.9558	0.7653	0.9310	0.7440
GCMC	0.9903	0.7761	0.9521	0.7479	0.9265	0.7297
FMG	1.0120	0.8170	1.0289	0.8264	1.0749	0.8394
HAN	1.6072	1.3002	1.6053	1.2187	1.6042	1.2175
CKE	1.0033	0.7841	1.0008	0.7817	0.9991	0.7795
KGAT	0.9821	0.7721	0.9445	0.7425	0.9181	0.7206
Ours(Linear)	0.9212	0.7297	0.9209	0.7295	0.9328	0.7373
Ours(Bilinear)	0.9049	0.7138	0.8993	0.7129	0.8976	0.7075
Improv.	**6.91%**	**7.55%**	**9.57%**	**3.99%**	**2.23%**	**1.82%**

5 EXPERIMENTS

In this section, we demonstrate the effectiveness of our model both quantitatively and qualitatively. We also perform ablation studies on the key designs of our model and study the impact of different meta-paths.

5.1 Experiments on MovieLens-1M Datasets

5.1.1 Datasets Description. The first dataset is the popular MovieLens-1M dataset[2]. MovieLens-1M dataset contains 3952 items and 6040 users, with 800,167 ratings ranging from 1 to 5. We select the items rated greater than 3 by a users as his/her positive items. For fair comparison with the knowledge-graph-based competitors, we only utilize the item side additional relationships. Thus, in terms of its network schema $(\mathcal{A}, \mathcal{R})$, objects type set $\mathcal{A}$ is comprised of user (short as **U**), movie(**M**) and movie type (**T**). The corresponding relation set is comprised of: user $\xrightarrow{rate}$ movie (**UM**), movie $\xrightarrow{belong\ to}$ type (**MT**). We follow [19] to design the following meta-paths: **UM** (user-movie feedbacks), **UMUM** (the user watched this movie also watched), **UMTM** (you might also like other this type of movies), **UMTMUM** (the user watched this type of movie also likes).

5.1.2 Implementation details. In M1 module, we build a simple two-layer MLP, with the number of neurons being 10 and 1, respectively. We adopt ELU functions as the activation function σ. In M2 module, we initialize the input feature matrix $\boldsymbol{X} = [\boldsymbol{X}^U || \boldsymbol{X}^I]$ as an identity matrix $\boldsymbol{I} \in \mathbb{R}^{(m+n)\times(m+n)}$, where the i-th row is a one-hot vector of node i's ID. The dimension k and d are set to 128 and 24 by validation, respectively. We determine the number of total time steps $\mathcal{T} = 2$ for all datasets empirically by observing that one or two steps of propagation serve well. We also adopt a dropout technique to both MLP module and the high-level projected features, with a dropout rate of 0.3. We adopt Adam optimizer [11] with a learning rate of 0.001 and weight decay of $5e-4$. As for the sampling procedure, for each mini-batch, we first sample 250 rating triplets. For each node i in the sampled rating triplets, we sample 25 neighboring nodes from its neighborhood $\mathcal{N}(i)$.

We randomly sample ratings into different training, validation and testing splits by a training ratio of 0.4/0.3/0.3, 0.6/0.2/0.2 and 0.8/0.1/0.1, respectively. Note that the path UM is set as the ratings between users and items. Hence, to assure the ratings for testing do not appear in the training phase, we built path UM only on training sets.

[2]https://grouplens.org/datasets/movielens/

Table 3: Performance Comparisons on Yelp.

Training Ratio	40%		60%		80%	
Metrics	RMSE ↓	MAE ↓	RMSE ↓	MAE ↓	RMSE ↓	MAE ↓
FM	1.1078	0.8956	1.1016	0.8739	1.1083	0.8743
GCMC	1.4676	1.2735	1.4076	1.2335	1.3676	1.1735
FMG	1.0972	0.8753	1.0893	0.8866	1.0889	0.8516
HAN	1.6182	1.1856	1.6183	1.1857	1.6305	1.1959
CKE	–	–	–	–	1.1189	1.0022
KGAT	–	–	–	–	1.0897	0.8848
Ours(Linear)	1.0888	0.8506	1.0755	0.8368	1.0714	0.8317
Ours(Bilinear)	1.0794	0.8425	1.0713	0.8335	1.0705	0.8267
Improv.	**1.62%**	**3.75%**	**1.65%**	**4.62%**	**1.69%**	**2.91%**

5.1.3 Competitors & Evaluation metrics. We compare against some state-of-the-art methods, which roughly cover using side information 1) as raw features: FM [16] and GCMC [27]; 2) to build meta-paths: FMG [32] and HAN [24]; 3) with KG-based methods: CKE [30] and KGAT [23].

We adopt the Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) metrics that are commonly used for rating prediction [17, 19].

5.1.4 Results analysis. The experimental results on MovieLens-1M are depicted in Table 2. The last rows of the tables are the relative improvements regarding the baseline FM. We can observe from tables that our models with both linear and bilinear recommendation modules achieve better performance against most competitors over three training-testing splits. Noteworthy, our bilinear model outperforms the second-best models on different training-testing splits by 8.03%, 4.68%, 3.04%, respectively over MAE, and by 6.91%, 5.55% and 3.13%, respectively over RMSE. This demonstrates the effectiveness of our proposed embedding-based recommendation framework.

One may notice that the performance of FMG on MovieLens is worse than FM which does not use meta-paths. However, the phenomenon does not exist on the other datasets. The reason is probably that the path-based similarities contain many noises and will even damage the recommendation performance if they are not effectively fused and utilized. On MovieLens-1M dataset, since both rating data and other relationship data are denser, the further combination of them may introduce much more noise. However, our method exhibit relatively stable improvements on all of the datasets consistently. The consistent performance may be contributed to that the nonlinear early fusion module first extracts important information out of complicated relationships among entities.

Table 4: Performance Comparisons on Amazon.

Training Ratio	40%		60%		80%	
Metrics	RMSE ↓	MAE ↓	RMSE ↓	MAE ↓	RMSE ↓	MAE ↓
FM	1.2026	0.9423	1.2066	0.9355	1.1877	0.9275
GCMC	1.5736	1.4870	1.5233	1.4263	1.4338	1.2873
FMG	1.1959	0.9257	1.1832	0.9253	1.1841	0.9031
HAN	1.3950	0.8512	1.2736	0.8603	1.4142	0.8732
CKE	1.3301	1.2236	1.3288	1.2322	1.3168	1.2102
KGAT	1.2551	0.9777	1.2547	0.9624	1.2422	0.9492
Ours(Linear)	1.3085	0.9811	1.2513	1.0007	1.2592	0.8636
Ours(Bilinear)	1.1823	0.8828	1.1861	0.8839	1.1979	0.8550
Improv.	**1.14%**	**3.71%**	–	–	–	**5.33%**

5.2 Experiments on Yelp Datasets

5.2.1 Datasets description. Yelp[3] is a business directory service and crowd-sourced review forum, where a user can rate businesses or post photos and reviews about them. In terms of its network schema $(\mathcal{A}, \mathcal{R})$, objects type set $\mathcal{A}$ is comprised of user (**U**), business (**B**), city (**Ci**), category (**Cat**). The relations are comprised of user $\xrightarrow{rate}$ business (**UB**), business $\xrightarrow{be\ located\ in}$ city (**BCi**), business $\xrightarrow{be\ categorized\ into}$ category (**BCat**). We follow the previous work [32] to design the following meta-paths: **UB** (user-item interactions), **UBUB** (the user who choose the same business also choose), **UBCiB** (other businesses that locate in the same city), **UBCatB** (the other businesses that share the same category with this business). We randomly select a subset that contains 5,413 users and 4,761 items with 17,802 ratings between them. The statistics are shown in Table 1.

5.2.2 Results analysis. The comparative results on Yelp are listed in Table 3. Because Yelp is highly sparse, the available users for two KG-based methods' implementations are few. Thus we do not attach the results of them on Yelp with training ratio of 0.4 and 0.6. As shown in the table, our models with both linear and bilinear recommendation models consistently outperform the other competitors over MAE and RMSE. Compared with the second-best models, our bilinear recommendation model gains relative improvements of 3.75%, 4.62% and 2.93% respectively over MAE, and of 1.63%, 1.65% and 1.69% over RMSE. Not considering its sensitiveness to data sparsity, GCMC indicates the effectiveness of neighborhood-based embedding methods for recommendation, which is consistent with the good performance of our model.

5.3 Experiments on Amazon Dataset

Dataset description. Amazon is an online shopping website where user-item ratings, user reviews about items are recorded. The objects in Amazon dataset [4] provided by [32] contain: users (**U**), businesses (**B**), brands (**Brd**), category (**Cat**). The relations are comprised of user $\xrightarrow{rate}$ business (**UB**), business $\xrightarrow{belong\ to}$ brand (**BBrd**), business $\xrightarrow{belong\ to}$ category (**BCat**).

We follow the previous work [32] to design the following meta-paths: **UB** (representing user-item interaction), **UBUB** (the users that have bought the same items also bought), **UBBrdB** (other items of the same brand), **UBCatB** (other items of the same category).

Results analysis. The experimental results on Amazon are depicted in Table 4. The last row is the relative improvements regarding the baseline FM. Our bilinear method exhibits top-2 performance. From the results, we notice that GCMC model, which also generates embeddings based on neighbors, performs poorly on both Yelp and Amazon. However, on MovieLens-1M, it performs well. The reason may be that GCMC learns the weight w_{uv} only from the rating data s_{uv}, thus could not learn a good fusion when rating data are severely sparse. Our model overcomes this issue since we learn the weights between neighbors not only from rating data but also from multiple meta-paths. The rich semantics behind meta-paths mitigate the data sparsity problem to some extent. Moreover, all

[3] https://www.yelp.com/dataset challenge
[4] http://jmcauley.ucsd.edu/data/amazon/

Table 5: The percentages in the brackets are the performance degradation ratio w.r.t our bilinear model.

Models	BiCutM1	BiCutSoftmax	Bilinear
RMSE	1.205 (−1.4%)	1.245 (−4.7%)	1.188
MAE	0.948 (−5.8%)	1.084 (−21%)	0.895

Table 6: Ablation study on M1 and softmax operation.

	MovieLens-1M		Yelp	
Models	RMSE	MAE	RMSE	MAE
BiCutM1	0.957 (-5.5%)	0.749 (-5.9%)	1.108 (-3.5%)	0.846 (-2.3%)
BiCutSoft	1.084 (-20%)	0.828 (-17%)	1.192 (-11%)	0.937 (-13%)
Billinear	0.898	0.708	1.071	0.827

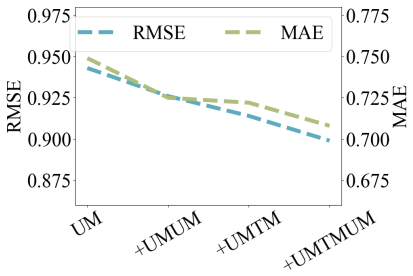


Figure 3: Performance changes when gradually adding meta-paths on MovieLens-1M.

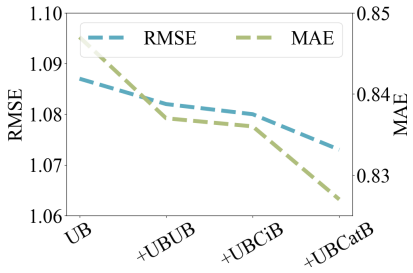
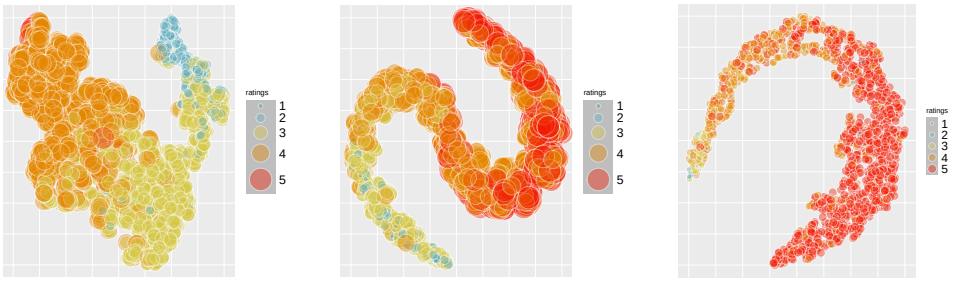


Figure 4: Performance changes when gradually adding meta-paths on Yelp.



(a) MovieLens (b) Yelp (c) Amazon

Figure 5: t-SNE projection of item embeddings. Different colors/sizes represent different average rating levels of items.

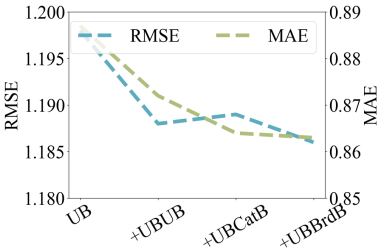


Figure 6: Performance changes when gradually adding meta-paths on Amazon.

the learnable parameters are shared among all nodes, which makes our model insensitive to data sparsity.

5.4 Ablation study

Flexible fusion module. To study the effectiveness of M1, we construct a model where α_{uv} and α_{vu} are replaced with a binary value 0 (if user u and item v do not have any observed interaction) or 1 (if user u and item v have an interaction). That is, the step to learn $\boldsymbol{\alpha}$ is skipped. We denote this model as *BiCutM1* and choose the original bilinear model as the baseline. The performance comparisons on MovieLens-1M are listed in Table 5. As can be seen from Table 5, there is an obvious performance degradation when M1 is removed. This demonstrates the effectiveness of the proposed M1 module to preserve useful information for recommendations.

Normalization operation. We construct a bilinear model where the normalization (Eq.3) is removed (denoted as *BiCutNorm*). The comparison results on MovieLens-1M are recorded in Table 6, from which we can observe a dramatic performance degradation over both RMSE and MAE. It indicates the necessity of the designed softmax operation to preserve asymmetry and achieve normalization simultaneously.

Impacts of different meta-paths. To further analyze the impact of different meta-paths, we provide an experiment here on different path sets by gradually adding one meta-path at a time. The performance changes are illustrated in the Figure 4, where we can observe that both RMSE and MAE are generally getting smaller, which means the performance improves as more meta-paths are incorporated.

Embedding visualization. To see the quality of embeddings for recommendation, we demonstrate the t-SNE projection of the final embeddings as shown in Fig. 5(a), 5(b), 5(c). The embeddings are the outputs of the last fully connected layer before the bilinear model. Different colors and sizes correspond to different rating levels (from 1 to 5), and each item's rating label is a discrete approximation of its averaged ground-truth ratings from all users. We rule out the items whose ratings given by different persons vary greatly (whose ratings' variance is bigger than 95% quantile of all items' ratings.)

From the figures, items' embeddings are gradually distributed on a manifold as average ratings range from 1 to 5. Different from the labels in classification tasks where each labels contain a specific type of semantics, the rating levels here, though varying in values, contain consistent semantics. Hence, no clusters appear in the visualization. Instead, nodes of different rating levels are gradually distributed and can be roughly distinguished. Most of the nodes rated 4 or 5 can be separated from those rated 1 or 2. This indicates our model does capture the useful semantics for recommendations.

6 CONCLUSION

This paper explores HIN-based recommendation problem with the help of meta-paths to help alleviate the sparsity of the user-item interactions in real world. We propose a flexible fusion module to flexibly assembling rich semantics of different meta-paths into the unified relative similarity graph between users and items. We then incorporate the fused semantics into user/item information propagation. After that, we design two schemes to conduct recommendations. Experimental results on three real-world datasets demonstrate the effectiveness of our model.

ACKNOWLEDGMENTS

This work was supported in part by the National Key R&D Program of China under Grant 2018AAA0102003, in part by National Natural Science Foundation of China: 61931008, 61620106009, 61836002 and 61976202, in part by the Fundamental Research Funds for the Central Universities, in part by Youth Innovation Promotion Association CAS, in part by the Strategic Priority Research Program of Chinese Academy of Sciences, Grant No. XDB28000000, in part by the National Postdoctoral Program for Innovative Talents under Grant BX2021298 and in part by MindSpore, which is a new deep learning computing framework [5].

[5]https://www.mindspore.cn

REFERENCES

[1] Ye Bi, Liqiang Song, Mengqiu Yao, Zhenyu Wu, Jianming Wang, and Jing Xiao. 2020. A Heterogeneous Information Network Based Cross Domain Insurance Recommendation System for Cold Start Users. In *SIGIR*. 2211–2220.

[2] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. 2019. Unifying Knowledge Graph Learning and Recommendation: Towards a Better Understanding of User Preferences. In *WWW*. 151–161.

[3] Yukuo Cen, Xu Zou, Jianwei Zhang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Representation Learning for Attributed Multiplex Heterogeneous Network. In *KDD*. 1358–1368.

[4] Jie Chen, Tengfei Ma, and Cao Xiao. 2018. FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling. In *ICLR*.

[5] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*. 1024–1034.

[6] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. 1989. Multilayer feedforward networks are universal approximators. *Neural Networks* 2, 5 (1989), 359–366.

[7] Binbin Hu, Chuan Shi, Wayne Xin Zhao, and Tianchi Yang. 2018. Local and Global Information Fusion for Top-N Recommendation in Heterogeneous Information Network. In *CIKM*. 1683–1686.

[8] Binbin Hu, Chuan Shi, Wayne Xin Zhao, and Philip S. Yu. 2018. Leveraging Meta-path Based Context for Top-N Recommendation with A Neural Co-Attention Model. In *KDD*. 1531–1540.

[9] Liang Hu, Songlei Jian, Longbing Cao, Zhiping Gu, Qingkui Chen, and Artak Amirbekyan. 2019. HERS: Modeling influential contexts with heterogeneous relations for sparse and cold-start recommendation. In *AAAI*, Vol. 33. 3830–3837.

[10] Jiarui Jin, Jiarui Qin, Yuchen Fang, Kounianhua Du, Weinan Zhang, Yong Yu, Zheng Zhang, and Alexander J Smola. 2020. An efficient neighborhood-based interaction model for recommendation on heterogeneous graph. In *KDD*. 75–84.

[11] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

[12] Hui Li, Yanlin Wang, Ziyu Lyu, and Jieming Shi. 2020. Multi-task Learning for Recommendation over Heterogeneous Information Network. *TKDE* 01 (2020), 1–1.

[13] Yuanfu Lu, Yuan Fang, and Chuan Shi. 2020. Meta-learning on Heterogeneous Information Networks for Cold-start Recommendation. In *KDD*. 1563–1573.

[14] Chen Luo, Wei Pang, Zhe Wang, and Chenghua Lin. 2014. Hete-cf: Social-based collaborative filtering recommendation using heterogeneous relations. In *ICDM*. 917–922.

[15] Weizhi Ma, Min Zhang, Yue Cao, Woojeong Jin, Chenyang Wang, Yiqun Liu, Shaoping Ma, and Xiang Ren. 2019. Jointly Learning Explainable Rules for Recommendation with Knowledge Graph. In *WWW*. 1210–1221.

[16] Steffen Rendle. 2012. Factorization Machines with libFM. *TIST* 3, 3 (2012), 57:1–57:22.

[17] Chuan Shi, Binbin Hu, Wayne Xin Zhao, and Philip S. Yu. 2019. Heterogeneous Information Network Embedding for Recommendation. *TKDE* 31, 2 (2019), 357–370.

[18] Chuan Shi, Jian Liu, Fuzhen Zhuang, S Yu Philip, and Bin Wu. 2016. Integrating heterogeneous information via flexible regularization framework for recommendation. *KAIS* 49, 3 (2016), 835–859.

[19] Chuan Shi, Zhiqiang Zhang, Yugang Ji, Weipeng Wang, S Yu Philip, and Zhiping Shi. 2019. SemRec:a personalized semantic recommendation method based on weighted heterogeneous information networks. In *WWW*. 153–184.

[20] Yizhou Sun and Jiawei Han. 2012. Mining heterogeneous information networks: a structural analysis approach. *SIGKDD Explorations* 14, 2 (2012), 20–28.

[21] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. 2011. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *VLDB Endowment* 4, 11 (2011), 992–1003.

[22] Xiao Wang, Deyu Bo, Chuan Shi, Shaohua Fan, Yanfang Ye, and Philip S. Yu. 2020. A Survey on Heterogeneous Graph Embedding: Methods, Techniques, Applications and Sources. *arXiv preprint arXiv:2011.14867* (2020).

[23] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge Graph Attention Network for Recommendation. In *KDD*. 950–958.

[24] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S. Yu. 2019. Heterogeneous Graph Attention Network. In *WWW*. 2022–2032.

[25] X. Wang, Y. Lu, Chuan Shi, Ruijia Wang, Peng Cui, and Shuai Mou. 2020. Dynamic Heterogeneous Information Network Embedding with Meta-path based Proximity. *arXiv preprint arXiv: 1701.05291* (2020).

[26] Zekai Wang, Hongzhi Liu, Yingpeng Du, Zhonghai Wu, and Xing Zhang. 2019. Unified embedding model over heterogeneous information network for personalized recommendation. In *IJCAI*. 3813–3819.

[27] Yuexin Wu, Hanxiao Liu, and Yiming Yang. 2018. Graph Convolutional Matrix Completion for Bipartite Edge Prediction. In *IC3K*. 49–58.

[28] Xiao Yu, Xiang Ren, Quanquan Gu, Yizhou Sun, and Jiawei Han. 2013. Collaborative filtering with entity similarity regularization in heterogeneous information networks. *IJCAI HINA* 27 (2013).

[29] Xiao Yu, Xiang Ren, Yizhou Sun, Quanquan Gu, Bradley Sturt, Urvashi Khandelwal, Brandon Norick, and Jiawei Han. 2014. Personalized entity recommendation: A heterogeneous information network approach. In *WSDM*. 283–292.

[30] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative Knowledge Base Embedding for Recommender Systems. In *KDD*. 353–362.

[31] Weina Zhang, Xingming Zhang, Haoxiang Wang, and Dongpei Chen. 2019. A deep variational matrix factorization method for recommendation on large scale sparse dataset. *Neurocomputing* 334 (2019), 206–218.

[32] Huan Zhao, Quanming Yao, Jianda Li, Yangqiu Song, and Dik Lun Lee. 2017. Meta-graph based recommendation fusion over heterogeneous information networks. In *KDD*. 635–644.

[33] Jing Zheng, Jian Liu, Chuan Shi, Fuzhen Zhuang, Jingzhi Li, and Bin Wu. 2017. Recommendation in heterogeneous information network via dual similarity regularization. *JDSA* 3, 1 (2017), 35–48.