Multi-task Learning for Recommendation over Heterogeneous Information Network

Hui Li, Yanlin Wang, Ziyu Lyu, and Jieming Shi

Abstract—Traditional recommender systems (RS) only consider homogeneous data and cannot fully model heterogeneous information of complex objects and relations. Recent advances in the study of Heterogeneous Information Network (HIN) have shed some light on how to leverage heterogeneous information in RS. However, existing HIN-based recommendation models assume HIN is invariable and merely use HIN as a data source for assisting recommendation, which limits their performance. In this paper, we propose a multi-task learning framework, called MTRec, for recommendation over HIN. MTRec relies on self-attention mechanism to learn the semantics of meta-paths in HIN and jointly optimizes the tasks of both recommendation and link prediction. Using a Bayesian task weight learner, MTRec is able to achieve the balance of two tasks during optimization automatically. Moreover, MTRec provides good interpretabilities of recommendation through a "translation" mechanism which is used to model the three-way interactions among users, items and the meta-paths connecting them. Experimental results demonstrate the superiority of MTRec over state-of-the-art HIN-based recommendation models, and the case studies we provide illustrate that MTRec enhances the explainability of RS.

Index Terms—Recommender Systems, Heterogeneous Information Network, Multi-task Learning.

1 Introduction

Traditional recommender systems (RS) rely on collaborative filtering (CF) methods, especially matrix factorization (MF) [1], to characterize the historical user-item interactions (i.e., ratings) and offer recommendations. However, CF-based approaches suffer from cold-start and data sparsity problems. The system lacks rich interaction information for many users/items, which downgrades its performance. With the development of web services, various types of data (e.g., text, image, location and social network) become available in RS [1] and researchers start to consider incorporating such auxiliary information into recommendation models in order to alleviate the cold-start and data sparsity problems. On the other hand, the rich side information raises new challenges for RS. Because of the heterogeneous nature of the real world, auxiliary data typically contains different objects with multiple types of interactions among them. Traditional recommendation approaches only consider homogeneous data and cannot fully model heterogeneous information in modern RS.

Due to the importance of heterogeneity in real-world applications, researchers have paid extensive attention to the study of heterogeneous information network (HIN). In HIN, objects are of different types, and links among objects represent different relations (i.e., interactions). Based on the fusion of heterogeneous information from HIN, various data mining tasks can be exploited, e.g., similarity search, clustering, classification and link prediction [2]. Recent advances in the study of HIN [2] have also shed some light on how to utilize heterogeneous data in RS and *HIN-based* RS have attracted much attention [3, 4, 5, 6, 7]. To better depict user preferences and item properties in RS, HIN-based recommendation approaches learn the semantic relations behind different types of interactions using meta-path [8] or meta-graph [9] (also called meta-structure [10]). The advantages of HIN-based recommendation models over traditional methods are twofold. Firstly, HIN-based recommendation methods are able to model the complex interactions between different objects. Characterizing and using such information is the key to improve the quality of recommendation since interactions among different objects affect users' behaviors, especially when users make purchasing decisions. Moreover, HIN-based recommendation approaches typically harness similarities based on meta-path/meta-graph [11] or the attention mechanism [6] to distinguish different parts of HIN (e.g., path or subgraph) according to their importance to users' behaviors. Consequently, HIN-based recommendation approaches provide a way to understand which piece of information affects recommendation results and are of great value to *Explainable Recommender Systems* [12].

Although HIN-based approaches have exceeded traditional models on both performance and interpretability, there are still some limitations of current HIN-based RS. A tremendous amount of work relies on leveraging meta-path/meta-graph based similarity [3, 11, 13, 14, 15] to improve the quality of recommendation. To obtain such similarity, HIN has to be represented as a graph structure, and then substructures like meta-path or meta-graph can be extracted from it, for the computation of the similarity. However, the high computational complexity [16] of similarity computation hinders these methods from being fully deployed in real large-scale HIN [7]. A compromise (e.g., limiting the length of meta-path [8]) has to be struck at the sacrifice of inaccurate similarity. Moreover, similarity based

- *Hui Li is with School of Informatics, Xiamen University, Xiamen, Fujian, China. E-mail: hui@xmu.edu.cn.*
- *Yanlin Wang is with Microsoft Research Asia, Beijing, China. E-mail: yanlwang@microsoft.com. She is the corresponding author.*
- *Ziyu Lyu is with Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, Guangdong, China. E-mail: zy.lv@siat.ac.cn.*
- *Jieming Shi is with School of Computing, National University of Singapore, Singapore. E-mail: shijm@nus.edu.sg.*

on meta-path/meta-graph may not fully reflect the latent features of users and items in HIN [7].

With the surge of research on network embeddings [16], some researchers start to consider utilizing node embeddings to overcome the limitations of similarity-based methods. Embedding based methods efficiently represent HIN in a low-dimensional space while preserving the semantic meanings. The node embeddings are either learned independently for subsequent use in MF [7] or constitute the representations in a neural network based model [6]. However, embedding based methods also have their limitations: (1) HIN is supposed to be *invariable* in these methods and the information flow between HIN and the recommendation module is monodirectional. In other words, HIN will not get updated in the lifecycle of RS, and it is solely used as a data source for assisting recommendation. (2) Moreover, HIN is keeping growing. The increase of the data in RS brings new user-item interactions and item-item relations to HIN. The enhanced HIN can further improve the recommendation. Therefore, there exists a mutual effect between HIN and recommendation. Nevertheless, existing approaches merely optimize recommendations, ignoring that there are other relevant tasks (e.g., modeling the dynamic growth of HIN) and learning multiple tasks jointly may further improve the generalization performance of all the tasks.

Aiming at solving the issues above, we propose a **M**ulti-**T**ask learning framework for **REC**ommendation over HIN (abridged as MTRec) in this paper. Our contributions are summarized as follows:

- We design a meta-path based recommendation model which mostly relies on the attention mechanism. By utilizing self-attention, the importance of each ingredient in the meta-path can be identified during the fine-grained learning.
- We model the meta-path as the "*translation*" from a user to an item in a heterogeneous way to depict the three-way interactions among users, items and the meta-paths connecting them. The translation based three-way modeling improves not only the performance but also the interpretability of the recommendation.
- For the first time, multi-task learning is introduced into HIN-based RS. We select link prediction, one task for modeling the dynamics of the network, as the auxiliary task and design a multi-task learning framework which optimizes both the recommendation task and the link prediction task in HIN.
- The balance of main and auxiliary tasks is automatically achieved using a Bayesian task weight learner during joint optimization. The Bayesian task weight learner can benefit not only the HIN-based recommendation task in this paper but also other tasks involving multi-task learning and negative sampling.

We conduct extensive experiments on real public data sets which show that MTRec has a superior performance compared to the state-of-the-art models on the task of HIN-based recommendation. Further, MTRec enhances the interpretability of recommendation via its "*translation*" mechanism and helps us better understand the recommendation results. Moreover, MTRec is able to provide predictions for the missing links in HIN, which is one task for modeling the dynamic growth of HIN. Therefore, MTRec is of high practical value compared to previous HIN-based recommendation methods which merely consider the recommendation task.

The rest of this paper is organized as follows. Sec. 2 provides the background. We present the details of MTRec in Sec. 3. Experiments on real data sets that demonstrate the effectiveness and the interpretability of MTRec are reported in Sec. 4. Sec. 5 discusses the related work. Sec. 6 concludes the paper and provides directions for future work.

Notation: We use lower-case fonts for scalars, bold lower-case fonts for vectors and bold upper-case font for matrices. For example, p is a scalar, $\mathbf{p}$ is a vector and $\mathbf{P}$ is a matrix.

2 PRELIMINARIES

In this section, we introduce relevant concepts used in this paper. A recommender system consists of m users and n items and their interactions are represented by a user-item interaction matrix $\mathbf{R} \in \mathbb{R}^{m \times n}$ with each non-zero element $r_{i,j} = 1$ indicating an observed interaction between user i and item j (e.g., i has rated or viewed j). Heterogeneous Information Network is defined as follows [8]:

Definition 1 (Heterogeneous Information Network). HIN is denoted as a directed graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ where $\mathcal{V}$ is an object set and $\mathcal{E}$ is a link set. A HIN is also associated with an object type mapping function $\phi : \mathcal{V} \rightarrow \mathcal{A}$ and a link type mapping function $\psi : \mathcal{E} \rightarrow \mathcal{R}$. $\mathcal{A}$ and $\mathcal{R}$ are sets of object types and link types where $|\mathcal{A} + \mathcal{R}| > 2$.

To describe the complex HIN, a network schema is provided for better understanding the structures and relations from meta level:

Definition 2 (Network Schema). Network schema, defined as $\mathcal{S} = \{\mathcal{A}, \mathcal{R}\}$, is a meta template for a HIN $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with object and link mapping functions ϕ and ψ.

In HIN, two objects can be linked through different semantic paths which are defined as meta-paths:

Definition 3 (Meta-path). A meta-path ρ is a path defined on a network schema $\mathcal{S} = \{\mathcal{A}, \mathcal{R}\}$, and is denoted in the form of $A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \cdots \xrightarrow{R_l} A_{l+1}$, which defines a composite relation $R = R_1 \circ R_2 \circ \cdots \circ R_l$ between object types A_1 and A_{l+1}, where $\circ$ represents the composition operator on relations.

Since there are multiple objects for each object type, a meta-path ρ can produce multiple meta-path instances. We use $h \in \rho$ to indicate that h is an instance of meta-path ρ. A meta-path (or meta-path instance) with a length of l indicates there are $l + 1$ object types (or objects) and l links.

By incorporating the idea of HIN into the recommendation, RS can better depict user preferences and item properties. We redefine the recommendation task in the setting of HIN:

Definition 4 (HIN-based Recommendation Problem). Given the user-item interaction matrix $\mathbf{R}$ and the corresponding HIN $\mathcal{G}$ in the system, for each user u, HIN-based RS aims to offer a recommendation list of k items that u is most interested in.

Some recent works have considered the rating prediction task in the setting of HIN-based RS [5]. Since top-k recommendation is more important and practical, we target at solving the problem defined in Definition 4.

Link prediction, which is a traditional data mining task, is introduced as an auxiliary task into MTRec in order to enhance the recommendation task under a multi-task architecture. Link prediction for HIN can be modeled from a perspective of sequential modeling, based on meta-paths. We give the definition of the auxiliary task as follows:

Definition 5 (Link Prediction for HIN). Each instance h of meta-path ρ, which has a length of l, can be segmented into l links: $h = \{\langle s_{\rho,h,1}, s_{\rho,h,2}\rangle, \langle s_{\rho,h,2}, s_{\rho,h,3}\rangle, \cdots, \langle s_{\rho,h,l}, s_{\rho,h,l+1}\rangle\}$, where $s_{\rho,h,b}$ is the b-th object in h. The link prediction problem for HIN can then be defined as a prediction problem of the o-th object in a meta-path instance, given its preceding $o-1$ objects.

3 MTRec

In this section, we introduce the proposed framework MTRec for multi-task learning in HIN-based RS.

3.1 Overview of Our Model

HIN is extremely spare, and there are many missing links (i.e., edges between different types of objects) in HIN-based RS for two reasons:

- There is a large number of items in the system. It is unlikely for one user to interact with most items. Consequently, a tremendous amount of links from users to other objects in HIN, which represent user interactions, are missing.
- Furthermore, a large part of the information for new items has not been collected in RS and the relations (i.e., links) from the new item to other objects do not appear in HIN.

The sparsity of HIN prevents it from assisting the recommendation task. Additionally, HIN is growing daily as users interact with items (e.g., view a webpage, buy a product or listen to music) and more information is collected by the system (e.g., product attributes). Assuming HIN is invariable when modeling HIN-based RS is unreasonable. Nevertheless, the sparsity and dynamics of HIN in HIN-based RS are commonly ignored in previous works.

To address these problems, we introduce an auxiliary task, link prediction for HIN, into MTRec and design the architecture of MTRec using multi-task learning. Link prediction of the network is a traditional data mining task. HIN-based recommendation task and link prediction for HIN task are relevant as they share some features of meta-paths. MTRec employs multi-task learning which introduces several benefits to HIN-based recommendation:

- Link prediction models the dynamic growth of HIN and it not only helps enrich the information of HIN (thus it alleviates the sparsity problem and information from meta-paths which do not bridge the target user and candidate items are also considered in MTRec), but also increases the performance of the recommendation task through sharing features and joint optimization in a multi-task manner.

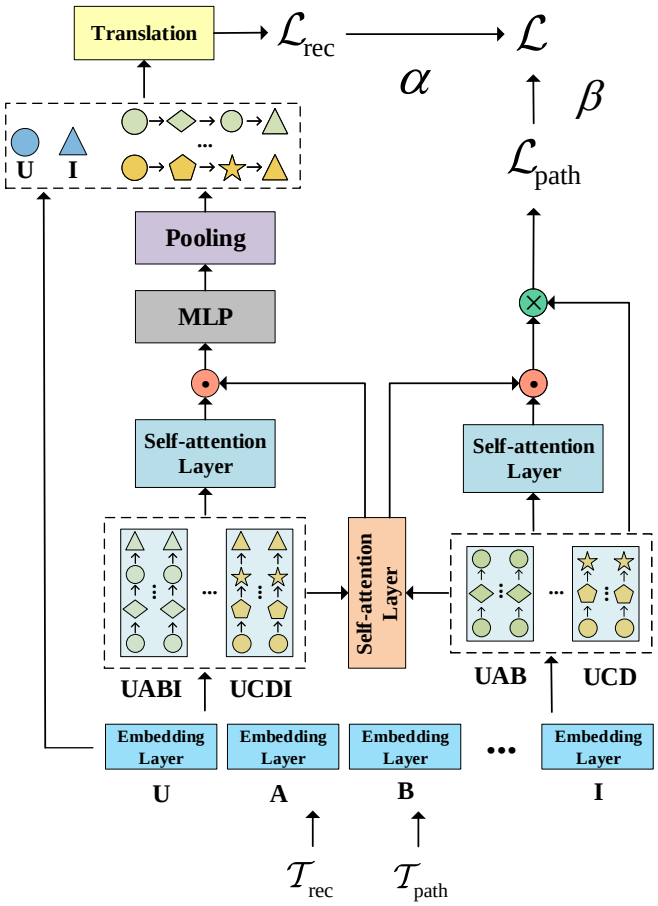

Fig. 1: Architecture of MTRec. "$\otimes$" is the operator of Hadamard product and "$\odot$" denotes the operator of concatenation. "U" and "I" represent users and items, respectively. "A", "B", "C" and "D" are four different object types.

- Multi-task learning helps the model to focus on most essential features, as the auxiliary task will provide additional evidence for the relevance or irrelevance of those features.
- Multi-task learning enhances the generalization ability of MTRec as it biases the model to prefer features that both the main task and the auxiliary task prefer.

Fig. 1 illustrates the overall architecture of MTRec that consists of two components. The left component (Sec. 3.2) is the self-attentive recommendation module, which provides top-k recommendations to users based on the meta-paths connecting users and candidate items. The right component (Sec. 3.3) is designed for the auxiliary task, link prediction in HIN, which is one of the tasks modeling the dynamics of HIN. The two modules share the underlying object embedding layer. Additionally, each component maintains one private feature space and shares a common feature space of meta-paths (Sec. 3.4.1) for multi-task learning. The private features in each component and the shared features are learned through the private self-attention layer in each component and a public self-attention layer, respectively. Finally, the two tasks are automatically balanced using a Bayesian task weight learner during the joint optimization (Sec. 3.4.2).

3.2 Main Task: HIN-based Recommendation

In this section, we will elaborate on the self-attentive recommendation module in MTRec.

3.2.1 Self-attentive Meta-path Modeling

Unlike previous embedding based approaches that leverage neural networks like CNN to learn the representations of meta-paths [6], MTRec mostly relies on a new concept, self-attention [17], to capture the complex semantics in meta-paths. Like other attention mechanisms which compute a weight distribution on the input sequence and assigns higher values to more relevant elements in the sequence [18], self-attention endows the model with the ability to focus on the most important parts of the sequential data and enhances the interpretation of the model. Differently, self-attention mechanism learns the attention weights by matching a sequence against itself. Inspired by the success of self-attention in various NLP tasks, especially machine translation [17], we adopt self-attention to model meta-paths in HIN-based RS which requires a sophisticated design as HIN-based recommendation is quite different from the machine translation task. Compared to most neural network based recommendation methods which use the attention mechanism as a supplemental component to enhance the original neural architecture (e.g., RNN or CNN), MTRec uses self-attention as its backbone in most part of the learning procedure.

The self-attentive meta-path modeling method used in MTRec can be decomposed into the following parts.

Object Embedding Layer. Since this paper focuses on the pure HIN-based CF setting, we only use the identity (i.e., ID) of each object in HIN as the input features. The input feature (e.g., the id of the object i with object type t) is represented as a binarized sparse vector $\mathbf{g}_i^{(t)}$ with one-hot encoding. Above the input layers are the embedding layers for all object types. Each embedding layer is a fully connected layer that projects the sparse one-hot representation $\mathbf{g}_i^{(t)}$ to a dense d-dimensional vector $\mathbf{s}_i^{(t)}$ to represent the corresponding object i with type t in HIN.

Meta-path Embedding Layer. As introduced in Sec. 2, each meta-path consists of a sequence of object types. We represent each meta-path instance by concatenating the embeddings of the objects in the sequence. Given an instance h of meta-path ρ with length $l-1$, its embedding $\mathbf{P}_{\rho,h} \in \mathbb{R}^{d\times l}$ is the stack of the embeddings for the objects in h:

$$\mathbf{P}_{\rho,h} = \begin{bmatrix} \mathbf{s}_{\rho,h,1}^{(1)} & \mathbf{s}_{\rho,h,2}^{(1)} & \cdots & \mathbf{s}_{\rho,h,l}^{(1)} \\ \vdots & \vdots & \cdots & \vdots \\ \mathbf{s}_{\rho,h,1}^{(d)} & \mathbf{s}_{\rho,h,2}^{(d)} & \cdots & \mathbf{s}_{\rho,h,l}^{(d)} \end{bmatrix} \tag{1}$$

where $\mathbf{s}_{\rho,h,o}^{(f)}$ indicates the f-th dimension of the o-th object in the meta-path instance h. Note that $\mathbf{s}$ is used to indicate object embedding in this paper with two usages: 1) $\mathbf{s}_{\rho,h,o}^{(f)}$ will be used when depicting its position in a meta-path; 2) When emphasizing the object id and object type, $\mathbf{s}_i^{(t)}$ will be used.

Self-attention Layer. A typical attention technique [18] maps a sequence of d-dimensional vectors, i.e., the *key* $\mathbf{K}$, to the attention weights $\mathbf{A}$. In this process, another input element $\mathbf{Q}$ called *query* is used as a reference and the attention technique computes the *compatibility* between the key and the query to give emphasis to the elements in the keys which are relevant to the query. Then, the compatibility is passed through a distribution function to obtain the attention weights $\mathbf{A}$. Finally, the attention weight is applied on another input element $\mathbf{V}$, the *value*, to obtain the refined representations of the key. In self-attention, the key, query and value refer to the same input sequence and this is how the name self-attention comes from.

In our model, the inputs of the key $\mathbf{K}$, the query $\mathbf{Q}$ and the value $\mathbf{V}$ are the same and compose of the meta-path instance embeddings $\mathbf{P}_{\rho,h}$. They are firstly projected to the same space with shared parameters:

$$\mathbf{K}_{\rho,h} = \mathbf{W}_K\mathbf{P}_{\rho,h},\ \mathbf{Q}_{\rho,h} = \mathbf{W}_Q\mathbf{P}_{\rho,h},\ \mathbf{V}_{\rho,h} = \mathbf{W}_V\mathbf{P}_{\rho,h} \tag{2}$$

where $\mathbf{W}_K, \mathbf{W}_Q, \mathbf{W}_V \in \mathbb{R}^{d\times d}$ are learnable weight matrices for $\mathbf{K}$, $\mathbf{Q}$ and $\mathbf{V}$, respectively.

Then, the affinity values $\mathbf{A}_{\rho,h} \in \mathbb{R}^{l\times l}$ (i.e., attention weights) can be obtained as follows:

$$\mathbf{A}_{\rho,h} = \text{softmax}\Big(\frac{\mathbf{K}_{\rho,h}^T \cdot \mathbf{Q}_{\rho,h}}{\sqrt{d}}\Big) \tag{3}$$

where $\sqrt{d}$ is used as the scaling factors to avoid the negative effects of small gradients. Each element $a_{c,e} \in \mathbf{A}$ reflects the relevance of the e-th object in the key sequence to the c-th object in the query sequence. It is worth noting that the key and the query is the same sequence in our model, i.e., one meta-path embedding. Consequently, the attention weight in Eq. 3 is computed by the self-matching of an input sequence.

After that, the refined instance embedding vector $\hat{\mathbf{p}}_{\rho,h}^T \in \mathbb{R}^d$ of instance h for meta-path ρ can be obtained by combining the value $\mathbf{V}$ and attention weights $\mathbf{A}$:

$$\begin{aligned} \hat{\mathbf{p}}_{\rho,h}^{(1)} &= \text{flatten}\big(\mathbf{V}_{\rho,h}\mathbf{A}_{\rho,h}\big) \\ \hat{\mathbf{p}}_{\rho,h}^{(2)} &= \tanh\big(\mathbf{W}_p\hat{\mathbf{p}}_{\rho,h}^{(1)} + \mathbf{b}_p\big) \end{aligned} \tag{4}$$

where "flatten$(\cdot)$" is the flattening operator which reshapes the $d \times l$ inputed matrix to $(d \times l)$-dimensional vector, $\mathbf{W}_p$ and $\mathbf{b}_p$ denote the weight matrix and the bias vector, and we use the hyperbolic tangent function as the activation function.

Our model further applies a mean pooling operation over the embeddings of all the meta-path instances for the meta-path ρ to derive the corresponding meta-path embedding $\hat{\mathbf{m}}_\rho \in \mathbb{R}^d$:

$$\hat{\mathbf{m}}_\rho = \text{mean}\,(\sum_{h\in\rho} \hat{\mathbf{p}}_{\rho,h}), \tag{5}$$

where "mean$(\cdot)$" is the mean pooling operation that extracts the average value of each dimension for embeddings of all meta-path instances.

After reviewing the whole process in self-attention layer, we can find that the position of each object in the meta-path instance does not affect the formulation for the refined meta-path embedding vector $\hat{\mathbf{m}}_\rho$. In other words, permuting the order of objects in one meta-path and the resulted meta-path will have the same representation as the original one.

To avoid such unreasonable representations, we inject a learnable position embedding $\mathbf{G} \in \mathbb{R}^{d\times l}$ into $\mathbf{P}_{\rho,h}$ before projecting it to $\mathbf{K}_{\rho,h}$, $\mathbf{Q}_{\rho,h}$ and $\mathbf{V}_{\rho,h}$ in Eq. 2 to help our model retain the sequential patterns of meta-path instances:

$$\mathbf{P}_{\rho,h} = \mathbf{P}_{\rho,h} + \mathbf{G}, \tag{6}$$

where each element in the i-th row of $\mathbf{G}$ is initialized with the value i. Through the injected positional embedding $\mathbf{G}$, our model is aware of the position of each object in the meta-path. Unlike [17] where a fixed position embedding is used, we find that applying a learnable position embedding makes the results more robust.

3.2.2 Modeling Three-way Interactions in Recommender

Most HIN-based RS, which rely on meta-paths, only characterize two-way user-item interactions, and seldom consider the mutual effect between the meta-path and the involved user-item pair in an interaction. Features learned from meta-paths are used to enhance user and item representations in recommendation models. Though these methods achieve performance improvements to some extent, they are unable to provide good interpretations about why meta-path and which meta-path helps improve the quality of recommendation due to the lack of modeling interactions among users, items and meta-paths connecting them [6].

To model the probability that a user i will accept an item j given the meta-paths connecting them, MTRec adopts the concept of "*translation*" which is prevalently used in *knowledge graph embedding* [19, 20]. In a knowledge graph, an edge can be represented as a triple $\langle$head entity, relation, tail entity$\rangle$ where head and tail entities are the nodes, and relation is the edge between them. Translation-based knowledge graph embedding [21] models the relation embedding $\mathbf{r}$ as the translation from head embedding $\mathbf{h}$ to tail embedding $\mathbf{t}$: $\mathbf{h}+\mathbf{r} \approx \mathbf{t}$. Inspired by the success of translation-based knowledge graph embedding approaches, we design a prediction layer for HIN-based recommendation with the idea that the meta-path can be viewed as the translation from the user to the item. The prediction layer for recommendation is defined as follows:

$$\begin{aligned} \mathbf{e}_{i,\rho,j} &= \text{relu}\Big(\mathbf{W}_\text{e}\big(\mathbf{s}_i^{(user)} + \hat{\mathbf{m}}_\rho - \mathbf{s}_j^{(item)}\big) + \mathbf{b}_e\Big) \\ r_{i,j} &= \sigma\Big(-\sum_{\rho\in\text{Path}(i,j)} ||\mathbf{e}_{i,\rho,j}||_2\Big) \end{aligned} \tag{7}$$

where $\mathbf{W}_e$ and $\mathbf{b}_e$ are the learnable parameters, $\text{Path}(i,j)$ denotes all the meta-paths connecting user i and item j, "$||\cdot||_2$" indicates the vector norm, $r_{i,j}$ indicates the likelihood that user i will interact with item j, and $\sigma(\cdot)$ is the sigmoid function.

If a meta-path ρ connecting a user i and an item j indicates a high likelihood that i will interact with j, then $\mathbf{s}_j^{(item)}$ should be close to $\mathbf{s}_i^{(user)} + \hat{\mathbf{m}}_\rho$. Different from translation-based knowledge graph embedding where the embeddings of head and tail entities belong to the same space, user embedding $\mathbf{s}_i^{(user)}$, meta-path embedding $\hat{\mathbf{m}}_\rho$ and item embedding $\mathbf{s}_j^{(item)}$ in Eq. 7 belong to three different vector spaces in MTRec. Therefore our model employs a more heterogeneous translation mechanism for modeling three-way interactions compared to translation-based knowledge graph embedding approaches.

One by-product of harnessing "*translation*" to model three-way interactions is the reinforcement for the interpretability of HIN-based recommendation. Observed from Eq. 7 that the L2 distance between the representation of a meta-path ρ (i.e., $\hat{\mathbf{m}}_\rho$) and $\mathbf{s}_j^{(item)} - \mathbf{s}_i^{(user)}$, where j is an item that user i is very likely to accept (i.e., $r_{i,j}$ is large), should be small if ρ plays a vital role in marking j as a positive candidate for the recommendation. Therefore, we can calculate the L2 distance between the representation of each meta-path connecting a ground-truth user-item pair and the corresponding representation $\mathbf{s}^{(item)} - \mathbf{s}^{(user)}$ (we call it *reference* representation in this paper) to explain which meta-path contributes to the recommendation most.

3.2.3 Loss Function of HIN-based Recommendation

We adopt negative sampling to train our model for the HIN-based recommendation task. Specifically, we uniformly sample n_rec items for each user i (i.e., $\text{neg}_\text{rec}(i)$) in the training set in each epoch. User i has not interacted with any item in $\text{neg}_\text{rec}(i)$ in the training set. We modify the binary cross entropy loss as the objective function:

$$\mathcal{L}_{rec}(\theta_{rec}) = -\sum_{\langle i,j\rangle\in\mathcal{T}_{rec}} \Big(\log r_{i,j} + \sum_{j'\in\text{neg}_\text{rec}(i)} \log\left(1 - r_{i,j'}\right)\Big), \tag{8}$$

where $\mathcal{T}_{rec}$ indicates the training data for the recommendation task, θ_{rec} is the parameters for the model, and $\langle i,j\rangle$ denotes that user i has interacted with item j.

3.3 Auxiliary Task: Link Prediction for HIN

Link prediction of a network is a tradition data mining task and various methods, either catered for HIN or not, can be adapted in our multi-task framework. Since we already harness self-attention mechanism to modeling the representations of meta-paths in HIN-based recommendation, we go along this direction in the task of link prediction for HIN (illustrated in Definition 5) for ease of exploration.

Similar to the recommendation module, we first obtain the meta-path instance embedding $\mathbf{P}_{\rho,h}$ for instance h of meta-path ρ and enhance it using the learnable positional embedding $\mathbf{G}$ as shown in Eq. 1 and Eq. 6. After that, $\mathbf{P}_{\rho,h}$ is passed through a self-attention layer to retrieve the refined embedding matrix $\hat{\mathbf{P}}_{\rho,h} \in \mathbb{R}^{d\times l}$ of the meta-path instance h:

$$\begin{aligned} \mathbf{A}_{\rho,h} &= \text{softmax}\Big(\frac{\mathbf{K}_{\rho,h}^T \cdot \mathbf{Q}_{\rho,h}}{\sqrt{d}}\Big) \\ \hat{\mathbf{P}}_{\rho,h} &= \mathbf{V}_{\rho,h}\mathbf{A}_{\rho,h} = \begin{bmatrix} \hat{\mathbf{s}}_{\rho,h,1}^{(1)} & \hat{\mathbf{s}}_{\rho,h,2}^{(1)} & \cdots & \hat{\mathbf{s}}_{\rho,h,l}^{(1)} \\ \vdots & \vdots & \cdots & \vdots \\ \hat{\mathbf{s}}_{\rho,h,1}^{(d)} & \hat{\mathbf{s}}_{\rho,h,2}^{(d)} & \cdots & \hat{\mathbf{s}}_{\rho,h,l}^{(d)} \end{bmatrix} \end{aligned} \tag{9}$$

where $\hat{\mathbf{s}}_{\rho,h,o}^{(f)}$ is the f-th dimension of the output representation for the o-th object. However, directly using $\hat{\mathbf{s}}_{\rho,h,o}$ as the features for predicting the o-th link will be erroneous [22]. Due to the nature of sequences, the model should consider only the first o objects when predicting the o-th link. However, we can observe that the o-th output of the self-attention layer contains embeddings of subsequent objects.

Therefore, we modified the attention mechanism in Eq. 9 for link prediction module by using masks to forbid all the links from $\mathbf{Q}_{\rho,h,o}$ to $\mathbf{K}_{\rho,h,f}$ where $f > o$. This way, $\hat{s}_{\rho,h,o}$ can be viewed as the features learned based on first o objects in an instance.

To predict the o-th link in a meta-path instance, we estimate the likelihood of each candidate object c for the $(o+1)$-th object based on the first o objects in the instance h. MTRec employs a product layer for the estimation:

$$q_{h,o+1,c} = \sigma(\hat{\mathbf{s}}^T_{\rho,h,o} \cdot \mathbf{s}_c), \tag{10}$$

where $\mathbf{s}_c$ is the embedding of the candidate object c that has the object type for the $(o+1)$-th object in the meta-path schema for ρ, and $\sigma(\cdot)$ is the sigmoid function.

3.3.1 Loss Function of Link Prediction.

Similar to the recommendation task, we employ negative sampling to cope with the optimization of the link prediction task. For each meta-path instance $h \in \rho$, we uniformly sample n_{path} objects (i.e., $\text{neg}_{\text{path}}(\rho, h, t)$) for each step t in h in each epoch. During sampling, any negative sample in $\text{neg}_{\text{path}}(\rho, h, t)$ should not be the direct subsequent object for the $(t-1)$-th object $s_{\rho,h,t-1}$ in any instance of ρ. We modify the binary cross entropy loss as the objective function:

$$\begin{aligned}\mathcal{L}_{path}(\theta_{path}) = -\sum_{\rho \in \mathcal{T}_{\text{path}}} \sum_{h \in \rho} \sum_{2 \le t \le |\rho|} \Big(& \log q_{h,t,c_t} \\ & + \sum_{j' \in \text{neg}_{\text{path}}(\rho,h,t)} \log\left(1 - q_{h,t,j'}\right) \Big),\end{aligned} \tag{11}$$

where θ_{path} are the parameters of the model for link prediction, $\mathcal{T}_{\text{path}}$ is the training data of link prediction task, $|\mathcal{T}_{path}|$ is the number of all meta-path instances, c_t is the id of the ground-truth object at t-th step of h, and $|\rho|$ indicates the number of object types contained in the meta-path ρ.

3.4 Multi-task Learning for HIN-based Recommender

In this section, we explain how MTRec can be optimized in a multi-task manner.

3.4.1 Feature Sharing

One key idea of multi-task learning is that features learned by different tasks should be divided into private and shared spaces, depending on whether parameters of some components should be shared. Fig. 2 depicts the private-shared model we use for parameter sharing between the HIN based recommendation task and the link prediction task.

For any meta-path instances h used either in recommendation task or link prediction task, its private representation and shared representation can be learned by passing it through a private self-attention layer in each task and a shared self-attention layer between two modules, respectively. Each self-attention layer is defined in Sec. 3.2. The final features for each task are the concatenation of private representation and shared representation.

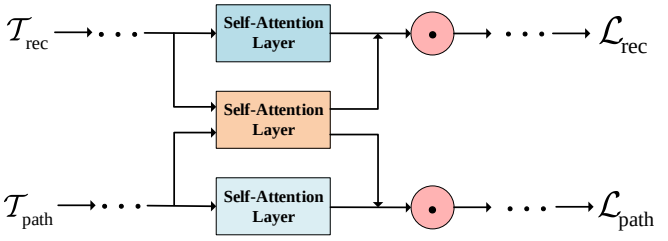

Fig. 2: Private-shared model for multi-task learning. "$\odot$" denotes the operator of concatenation.

3.4.2 Bayesian Task Weight Learner

Multi-task learning concerns the problem of optimizing a model with respect to multiple objectives. A common way to combine loss objectives of the two tasks, i.e., HIN based recommendation and link prediction, is to perform a weighted sum of the individual loss objective:

$$\mathcal{L} = w_{\text{rec}}\mathcal{L}_{\text{rec}} + w_{path}\mathcal{L}_{path}. \tag{12}$$

The harmony of the two tasks can be achieved through setting different weight values of w_{rec} and w_{path} for the two tasks. A naive but prevalent way is to set the weights to be equal for individual tasks. However, this workaround is only valid when the tasks do not compete, which is rarely the case. Different tasks need to be properly balanced so that network parameters converge to robust shared features that are useful across all tasks. Task imbalances impede proper training because they manifest as imbalances between back-propagated gradients. A task that is too dominant during training, for example, will express that dominance by inducing gradients which have relatively large magnitudes. Searching and dynamically updating optimal weights during optimization is a difficult and expensive process.

Inspired by the recent study which uses uncertainty to weigh losses in multi-task learning [23, 24], we leverage Bayesian modeling to derive the joint multi-task loss function and design a Bayesian task weight learner which can automatically achieve the balance between the two tasks.

Similar to Kendall et al. [24], we introduce an assumption $\frac{1}{\lambda^2}\left(\exp(\frac{x}{\lambda^2}) + 1\right) \approx \left(\exp(x) + 1\right)^{\frac{1}{\lambda^2}}$ which becomes an equality when $\lambda \to 1$. Based on the assumption, we can derive the following approximations for the sigmoid function $\sigma(\cdot)$:

$$\begin{aligned}\sigma(\frac{x}{\lambda^2}) &= \frac{\exp(\frac{x}{\lambda^2})}{\exp(\frac{x}{\lambda^2}) + 1} \approx \frac{1}{\lambda^2}\left(\frac{\exp(x)}{\exp(x)+1}\right)^{\frac{1}{\lambda^2}} = \frac{1}{\lambda^2}\left(\sigma(x)\right)^{\frac{1}{\lambda^2}} \\ 1 - \sigma(\frac{x}{\lambda^2}) &= \frac{1}{\exp(\frac{x}{\lambda^2}) + 1} \approx \frac{1}{\lambda^2}\left(\frac{1}{\exp(x)+1}\right)^{\frac{1}{\lambda^2}} = \frac{1}{\lambda^2}\left(1 - \sigma(x)\right)^{\frac{1}{\lambda^2}}\end{aligned} \tag{13}$$

Now let us investigate the loss function shown in Eq. 8 for the HIN-based recommendation. Let $x(i,j)$ be a user-item interaction $\langle i,j \rangle$ and $y(i,j)$ represents the label for $x(i,j)$ (1 if $\langle i,j \rangle$ is positive, otherwise 0). $R(x)$ indicates the outputs of MTRec for all x using Eq. 7 and $r_{x(i,j)}$ is the probability of a specific interaction $\langle i,j \rangle$ being positive. Then the binary classification likelihood can be defined as

follows:

$$\begin{aligned} Pr(y|R(x)) &= \prod_{\langle i,j\rangle \in \mathcal{T}_{rec}} Pr(y(i,j)|r_{x(i,j)}) \\ &= \prod_{\langle i,j\rangle \in \mathcal{T}_{rec}} \left(\sigma(r_{x(i,j)}) \prod_{j' \in \text{neg}_{\text{rec}}(i)} \left(1 - \sigma(r_{x(i,j')})\right)\right). \end{aligned} \tag{14}$$

Following Kendall et al. [24], we introduce a scalar α into Eq. 14 to get a *scaled* version of the model output:

$$Pr\Big(y\Big|R(x),\alpha\Big) = \prod_{\langle i,j\rangle \in \mathcal{T}_{rec}} \left(\sigma\big(\frac{r_{x(i,j)}}{\alpha^2}\big) \prod_{j' \in \text{neg}_{\text{rec}}(i)} \left(1 - \sigma\big(\frac{r_{x(i,j')}}{\alpha^2}\big)\right)\right), \tag{15}$$

which can be interpreted as a Boltzmann distribution (i.e., Gibbs distribution) [24]. The input is scaled by α^2 (often referred to as *temperature*). Then the log likelihood can be written as:

$$\begin{aligned} &\log Pr\Big(y\Big|R(x),\alpha\Big) \\ &= \sum_{\langle i,j\rangle \in \mathcal{T}_{rec}} \left(\log\Big(\sigma\big(\frac{r_{x(i,j)}}{\alpha^2}\big)\Big) + \sum_{j' \in \text{neg}_{\text{rec}}(i)} \log\Big(1 - \sigma\big(\frac{r_{x(i,j')}}{\alpha^2}\big)\Big)\right) \\ &\approx \sum_{\langle i,j\rangle \in \mathcal{T}_{rec}} \left(\frac{1}{\alpha^2}\log\Big(\sigma(r_{x(i,j)})\Big) + \frac{1}{\alpha^2}\sum_{j' \in \text{neg}_{\text{rec}}(i)} \log\Big(1 - \sigma(r_{x(i,j')})\Big) \right. \\ &\quad \left. - 2(n_{rec}+1)\log\alpha \right) \\ &= -\frac{1}{\alpha^2}\mathcal{L}_{rec}(\theta_{rec}) - 2(n_{rec}+1)\cdot|\mathcal{T}_{rec}|\cdot\log\alpha, \end{aligned} \tag{16}$$

where we introduce the approximations in Eq. 13 to the penultimate transition.

Similarly, we can obtain the log likelihood for the link prediction task with a scalar β, since the objective for link prediction in Eq. 11 has a similar form as the objective of recommendation in Eq. 8:

$$\log Pr\Big(y\Big|Q(x),\beta\Big) \approx -\frac{1}{\beta^2}\mathcal{L}_{path}(\theta_{path}) - 2(n_{path}+1)\cdot|\mathcal{T}_{path}|\cdot\bar{l}\cdot\log\beta, \tag{17}$$

where $Q(x)$ indicate the outputs of MTRec on the inputs x to the link prediction task using Eq. 10, y is the labels, $|\mathcal{T}_{path}|$ is the number of all meta-path instances, and $\bar{l}$ is the average length of meta-paths.

We maximize the log likelihood of MTRec in maximum likelihood inference, while we minimize the joint objective during optimization. Thus, the joint loss $\mathcal{L}$ can be formulated as:

$$\begin{aligned} \mathcal{L}(\theta_{rec},\theta_{path}) &= -\log Pr(y|U(x)) \\ &= -\log\Big(Pr(y_{rec}|R(x_{rec}),\alpha)\cdot Pr(y_{path}|Q(x_{path}),\beta)\Big) \\ &= \frac{1}{\alpha^2}\mathcal{L}_{rec}(\theta_{rec}) + \frac{1}{\beta^2}\mathcal{L}_{path}(\theta_{path}) \\ &\quad + 2(n_{rec}+1)\cdot|\mathcal{T}_{rec}|\cdot\log\alpha \\ &\quad + 2(n_{path}+1)\cdot|\mathcal{T}_{path}|\cdot\bar{l}\cdot\log\beta, \end{aligned} \tag{18}$$

where $U(x) = \{R(x_{rec}), Q(x_{path})\}$ is the outputs of MTRec on the inputs $x = \{x_{rec}, x_{path}\}$, and $y = \{y_{rec}, y_{path}\}$ is the labels.

The joint objective can be seen as learning the relative weights of the two losses. Small value of α will decrease the contribution of $\mathcal{L}_{rec}$, whereas large value will increase its contribution. β has a similar impact on the contribution of $\mathcal{L}_{path}$. Scales are regulated by the last two terms in Eq. 18. This way, we provide an automatic mechanism to balance the two tasks. Unlike Eq. 12 where the relative weights have to be set manually, the weights α and β in the joint loss of Eq.18 are automatically learned during optimization.

The Bayesian task weight learner is different compared to previous works [23, 24], which introduce the idea of uncertainty to automatically learn the task weights to balance a regression task and a classification task *without the negative sampling strategy*. It is nontrivial to design such a task weight learner without the approximations we proposed. Negative sampling strategy is a common training method used in many machine learning tasks including general recommendation [25] and knowledge graph embedding [19, 20]. The Bayesian task weight learner can benefit not only the HIN-based recommendation task in this paper but also other tasks involving multi-task learning and negative sampling.

4 Experiments

In this section, we conduct an experimental study using real data sets to show the effectiveness of MTRec.

4.1 Experimental Settings

4.1.1 Data

We use data sets MovieLens[1], LastFM[2] and Yelp[3] which are widely used in previous studies about HIN-based recommendation [6].

For the recommendation task, our evaluation focuses on implicit feedbacks. LastFM contains users' listening records which can be transformed as implicit feedbacks and directly used for our evaluation. For data sets MovieLens and Yelp, we follow [6, 25] and treat a user-item rating as an interaction record. We select the same meta-paths as [6]. These meta-paths contain at most 4 hops since long meta-paths are likely to introduce noisy semantics [8]. Tab. 1 explains the meaning of the notations used for these meta-paths. For example, relations "User-Movie" and "Movie-Genre" are denoted as "um" and "mg" in the following, respectively. Thus, the meta-path "umgm" in MovieLens indicates "User-Movie-Genre-Movie".

For assessing the quality of HIN-based recommendation task, we randomly selected 80% user-item interactions in each data set to be used for training; the remaining 20% interactions are held out for testing. We then use the methodology in [6] to select meta-path instances connecting users and items:

1) Firstly, user-item interaction matrix is factorized by CCDPP[4] [26], a matrix factorization method, to obtain user and item feature vectors.
2) Based on user and item feature vectors, we compute the top-50 most similar users/items for each user/item and generate {uu, mm}, {uu, aa} and {uu} relations for MovieLens, LastFM and Yelp, respectively. We adopt Peason's coefficient as the similarity measure. Other relations exist in the original data. Given these relations (i.e., edges), HIN can be constructed.

1. https://grouplens.org/datasets/movielens/100k/
2. https://grouplens.org/datasets/hetrec-2011/
3. https://www.kaggle.com/c/yelp-recsys-2013
4. https://github.com/Hui-Li/CCDPP

3) We leverage HIN2Vec[5] [27], a representation learning method for HINs, to retrieve representations for each object in HIN.
4) Priority based random walk [6] is used to retrieve meta-paths connecting user-item pairs which have interaction records in the training data. The next object to visit is selected based on the similarity between the representations of the current object and all the out-going similar objects. Similarity threshold is set to be 0.9 for all data. Additionally, we use a threshold of 0.8 to generate a larger version of Yelp data which includes more meta-path instances. We denote it by Yelp-L.

For evaluating the link prediction task, we also adopt the priority based random walk to sample meta-path. Differently, the start object (user) and the end object (item) of the sampled instance are not required to constitute a user-item interaction in the data. Then, 80% meta-path instances are randomly sampled for training and the remaining instances are used for evaluation.

In addition to the explanation of notations, Tab. 1 shows the statistics for all data sets used in both main task and auxiliary task. It is worth noticing that the meta-path instances used in the main task is not the subset of the instances used in the auxiliary task, though they are overlapping. Meta-path instances for the auxiliary task may connect some user-item pairs in which the user has not interacted with the item, while instances for the main task connecting user-item interactions.

4.1.2 Baselines

We compare the performance of the following approaches:

- **BPR** [28] is the Bayesian Personalized Ranking model that minimizes the pairwise ranking loss for implicit feedback
- **NeuCF** [25] is the Neural Collaborative Filtering which utilizes implicit feedback for top-k recommendation.
- **NGCF** [29] is the Neural Graph Collaborative Filtering method which exploits the user-item graph structure by propagating embeddings on it.
- **VAES** [30] is the Variational Autoencoders based Collaborative Filtering method, which generalizes linear latent-factor models to a non-linear probabilistic latent-variable model.
- **NeuACF** [31] is the Aspect-Level Deep Collaborative Filtering approach which learns the aspect-level latent factors through different meta-paths and fuse them with an attention mechanism for top-k recommendation.
- **FMG** [3] is a meta-graph based method for rating prediction in recommender system. Similar to [6], we modify its optimization objective as pairwise ranking loss used in BPR [28] for top-k recommendation.
- **HERec** [7] is a HIN based recommendation method which adopts a meta-path based random walk strategy to generate meaningful object sequences for network embedding. The embeddings are then fused and integrated into a MF method for recommendation.
- **MCRec** [6] is a meta-path based model, which constructs a three-way neural interaction architecture for recommendation. CNN is used to extract meta-path embeddings and a co-attention mechanism is leveraged for modeling interactions among users, items and meta-paths.
- **MTRec**$_{rec}$ is our proposed approach for recommendation task in HIN. It only contains the method introduced in Sec. 3.2.
- **MTRec**$_{lp}$ is the method introduced in Sec. 3.3 for the link prediction task in HIN.
- **MTRec** is the complete framework for multi-task recommendation in HIN. It optimizes both the tasks of HIN-based recommendation and link prediction in HIN. Feature sharing and weighted loss illustrated in Sec. 3.4 are used in the framework.
- **MTRec**$_n$ is similar to MTRec, except that a *naive* weighted sum loss (Eq. 12) is used for multi-task learning. w_{rec} and w_{path} are set to be equal.

The learned representations (i.e., the output from Step 3 when selecting meta-path instances for evaluating HIN-based recommendation) for each object in HIN by HIN2Vec is utilized as the pre-trained features for MCRec. Our approach and BPR were implemented using PyTorch. Implementations for other methods are provided by their authors.

4.1.3 Evaluation

For a fair comparison, we follow the evaluation settings in [6] for the recommendation task. For each user-item pair in the test set, we randomly sample n_{rec} negative test items. The target user has not interacted with these negative items before. For each user in the test set, we use all the items he/she has interacted with in the test set and the corresponding negative items to construct a test list. Then we rank the list and the results are evaluated using Precision at rank k (Prec@k), Recall at rank k (Recall@k) and Normalized Discounted Cumulative Gain at rank k (NDCG@k).

For the auxiliary task, link prediction, we use a similar evaluation protocol as the recommendation task. For each meta-path instance with length $l-1$ in the test set, we predict the l-th object in the sequence based on the first $l-1$ objects. For each positive last object in an instance $h \in \rho$, we sample n_{path} negative objects which can not be the direct subsequent objects for the $(l-1)$-th object (i.e., $s_{\rho,h,l-1}$) in any instance of ρ. For each user in the test set, we use all the meta-path instances starting from him/her in the test set and the corresponding negative objects to construct a test list. We adopt Hit Ratio at rank k (HR@k), NDCG@k and Mean Reciprocal at rank k (MRR@k) which are the measures commonly used for sequential analysis [32] for evaluating link prediction task.

Hyperparameters. For all methods, we set learning rate, dimensionality d, training batch size, n_{rec} and n_{path} to 0.001, 128, 256, 20 and 100, respectively. Hypeparameters of each baselines are set as suggested by their authors. All methods are optimized using Adaptive Moment Estimation (Adam) [33] and trained until convergence.

4.2 Experimental Results

4.2.1 Performance for HIN based Recommendation

Overall Performance. Tab. 2 illustrates the performance of all methods for the recommendation on the four data sets when evaluating using rank $k = 10$. The first row of

5. https://github.com/csiesheep/hin2vec

TABLE 1: Statistics of the data

Data	Meta-Path Instance			Relations (A-B)	#A	#B	#A-B	
	Meta-Path	Main Task	Auxiliary Task				Main Task	Auxiliary Task
MovieLens	umgm	622,579	3,795,480	User-Movie (um)	943	1,682	8,159	152,261
	umum	82,386	3,948,204	User-User (uu)	943	943	39,878	33,250
	uuum	116,591	3,939,422	Movie-Movie (mm)	1,682	1,682	21,830	81,709
	ummm	566,959	3,949,972	Movie-Genre (mg)	943	18	3,188	5,401
LastFM	uata	1,200,800	7,571,725	User-Artist (ua)	1,892	17,632	1,561	127,761
	uaua	5,727	7,599,186	User-User (uu)	1,892	1,892	67,933	94,300
	uuua	44,641	7,593,927	Artist-Tag (at)	17,632	9,718	52,615	155,325
	uua	9,543	3,464,400					
Yelp	ubcb	95,692	11,039,997	User-Business (ub)	45,981	11,537	9,031	360,242
	ubib	1,900,681	10,947,136	User-User (uu)	45,981	45,981	4,003	2,100,970
	ubub	143,546	11,039,943	Business-City (bi)	11,537	53	10,972	17,637
	uub	15,033	4,538,526	Business-Category (bc)	11,537	449	3,341	46,458
Yelp-L	ubcb	456,590	29,439,828	User-Business (ub)	45,981	11,537	49,852	367,747
	ubib	6,616,949	28,872,769	User-User (uu)	45,981	45,981	7,570	2,107,482
	ubub	3,145,281	29,406,674	Business-City (bi)	11,537	57	11,938	18,809
	uub	106,032	4,559,795	Business-Category (bc)	11,537	468	16,736	49,099

TABLE 2: Performance of different methods for the recommendation task on four data sets

Method	MovieLens			LastFM			Yelp			Yelp-L		
	Prec@10	Recall@10	NDCG@10	Prec@10	Recall@10	NDCG@10	Prec@10	Recall@10	NDCG@10	Prec@10	Recall@10	NDCG@10
BPR	0.2171	0.2854	0.6351	0.5114	0.5256	0.8271	0.1464	0.7666	0.6555	0.1923	0.7412	0.6341
NeuCF	0.2214	0.2531	0.6287	0.5648	0.5412	0.8348	0.1537	0.7751	0.6781	0.2122	0.7555	0.6587
NGCF	0.2693	0.2712	0.6211	0.6173	0.5745	0.8791	0.1943	0.8097	0.7815	0.2045	0.7234	0.6612
VAES	0.2811	0.2907	0.6618	0.6011	0.5891	0.8634	0.1922	0.7898	0.6912	0.2234	0.7512	0.6678
NeuACF	0.2459	0.2487	0.6597	0.5878	0.5587	0.8387	0.1578	0.7812	0.6698	0.2123	0.7511	0.6513
FMG	0.2618	0.2780	0.6978	0.6189	0.5987	0.8810	0.1723	0.7847	0.6787	0.2490	0.7798	0.6712
HERec	0.2421	0.2578	0.6345	0.5745	0.5542	0.8478	0.1482	0.7648	0.6878	0.2132	0.7423	0.6788
MCRec	0.2645	0.2677	0.6406	0.6148	0.5874	0.8950	0.1797	0.7779	0.6904	0.2367	0.7632	0.6811
MTRec$_{rec}$	0.3123	0.2978	0.7214	0.6478	0.6213	0.9178	0.2021	0.8064	0.8016	0.2722	0.7922	0.7532
MTRec$_n$	0.3321	0.3047	0.7310	0.6497	0.6311	0.9187	0.2098	0.8214	0.8097	0.2812	0.8001	0.7654
MTRec	**0.3431**	**0.3245**	**0.7354**	**0.6611**	**0.6478**	**0.9345**	**0.2154**	**0.8315**	**0.8214**	**0.2975**	**0.8193**	**0.7912**
Improve	22.06%	11.63%	5.39%	6.82%	8.20%	4.41%	10.86%	2.69%	5.11%	19.48%	5.07%	16.17%
	11.10%	2.44%	3.38%	4.67%	3.77%	2.55%	4.01%	-0.41%	2.57%	9.32%	1.59%	10.59%
	9.86%	8.97%	1.94%	2.05%	4.27%	1.82%	6.58%	3.11%	2.47%	9.29%	3.42%	5.05%
	3.31%	6.50%	0.60%	1.75%	2.65%	1.72%	2.67%	1.23%	1.44%	5.80%	2.40%	3.37%

TABLE 3: Performance of MTRec$_{lp}$, MTRec$_n$ and MTRec for the link prediction task on four data sets

Method	MovieLens			LastFM			Yelp			Yelp-L		
	HR@10	NDCG@10	MRR@10	HR@10	NDCG@10	MRR@10	HR@10	NDCG@10	MRR@10	HR@10	NDCG@10	MRR@10
MTRec$_{lp}$	4.14%	0.0253	0.0204	15.12%	0.1023	0.0870	23.28%	0.2008	0.1925	26.87%	0.3111	0.2122
MTRec$_n$	6.29%	0.0541	0.0410	18.65%	0.1248	0.1014	27.99%	0.2587	0.2214	29.11%	0.3465	0.2490
MTRec	**7.23%**	**0.0647**	**0.0498**	**19.17%**	**0.1339**	**0.1157**	**28.97%**	**0.2777**	**0.2478**	**32.44%**	**0.3651**	**0.2777**
Improve	74.64%	155.73%	144.12%	26.79%	30.89%	32.99%	24.44%	38.30%	28.73%	20.73%	17.36%	30.87%

'*Improve*' shows the improvement percentage of our model MTRec over other methods (except MTRec$_{rec}$ and MTRec$_n$) with the best performance on each evaluation measure, the second row demonstrates the improvement percentage of our model MTRec$_{rec}$ over other methods (except MTRec and MTRec$_n$) with the best performance on each evaluation measure, the third row depicts the improvement percentage of MTRec over MTRec$_{rec}$, and the forth row shows the improvement percentage of MTRec over MTRec$_n$.

From Tab. 2, we can observe that MTRec significantly outperforms existing recommendation approaches, either leveraging HIN or not, for the recommendation task. Compared to methods which do not utilize HIN (i.e., BPR, NeuCF, NGCF and VAES), approaches using HIN (i.e., NeuACF, FMG, HERec, MCRec, MTRec$_{rec}$, MTRec$_n$ and MTRec) show better performance in most cases, which demonstrates the usefulness of HIN in improving the quality of recommendation.

To investigate whether each component of MTRec contributes to the improvement of performance, we evaluate the performance of MTRec$_{rec}$. As shown in Tab. 2, it achieves a noticeable improvement compared to baselines. In the worst case, MTRec$_{rec}$ shows a comparable performance to the best baseline. In most cases, MTRec$_{rec}$ outperforms all the baselines significantly. From the results, we can conclude that the self-attentive meta-path modeling and three-way interaction modeling, as introduced in Sec. 3.2, play a vital role in increasing the performance of the recommender. On the other hand, the better results of MTRec$_{rec}$ shows that a pure attention based recommendation model can yield a comparable or even better recommendation compared to previous methods which rely on convolutional neural network (MCRec), multi-layer perceptron (NeuACF), graph neural network (NGCF) and variational autoencoder (VAES).

From Tab. 2, we can also find that introducing multi-task learning into MTRec help improve its performance for the recommendation task, since MTRec largely exceeds

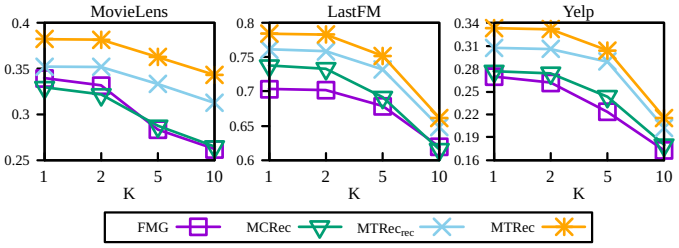

Fig. 3: Performance of different methods on Prec@k ($k = \{1, 2, 5, 10\}$) for the recommendation task

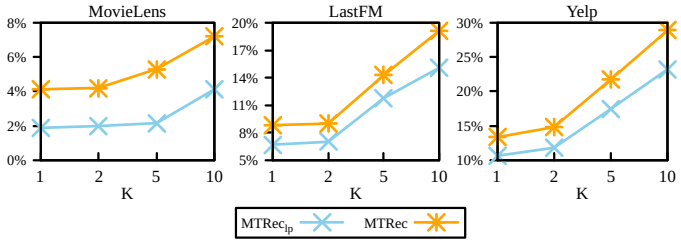

Fig. 4: Performance of different methods on HR@k ($k = \{1, 2, 5, 10\}$) for the link prediction task

MTRec$_{rec}$. This is consistent with our motivation of MTRec which is illustrated in Sec. 1: considering the auxiliary task in HIN during optimization can enhance the performance of the model on the recommendation task.

Performance on Different Rank k. To assess the robustness of our models, we report the Prec@k of HIN-based recommendation models (FMG, MCRec, MTRec$_{rec}$ and MTRec) on three data sets using $k = \{1, 2, 5, 10\}$ in Fig. 3. For other measures and Yelp-L data, similar trends can be observed. From Fig. 3, we can conclude that MTRec consistently outperforms other HIN-based recommendation methods which shows the robustness of MTRec.

4.2.2 Performance for Link Prediction in HIN

Overall Performance. The focus of this paper is improving the performance of recommendation and link prediction is the auxiliary task which assists the main task. For ease of exposition, we only compare the performance of MTRec$_{lp}$ and MTRec to show that the auxiliary task also benefits from the multi-task learning.

We report the results of link prediction on four data sets in Tab. 3 when evaluating using rank $k = 10$. The row of '*Improve*' shows the improvement percentage of MTRec over MTRec$_{lp}$ which only considers the link prediction task.

From Tab. 3 we can observe that our model MTRec, which considers both recommendation task and link prediction task, significantly surpasses its downgraded version MTRec$_{lp}$. From the results, we can draw the conclusion that the multi-task mechanism we design not only improves the performance on the main task but also benefits the auxiliary task.

Performance on Different Rank k. We also report the results of MTRec$_{lp}$ and MTRec using $k = \{1, 2, 5, 10\}$ for HR@k on three data sets to evaluate the robustness of our models. For other measures and Yelp-L data, the two models show similar trends. Fig. 4 illustrates the results. From Fig. 4, we can find that the performance of MTRec surpasses MTRec$_{lp}$ for different values of k on HR@k. Hence, both MTRec$_{lp}$ and MTRec are robust for the auxiliary task.

4.2.3 Effectiveness of the Bayesian Task Weight Learner

We also compare the performance of using naive joint loss (MTRec$_n$) and employing Bayesian weighted loss (MTRec). The evaluation results for recommendation task and link prediction task can be found in Tab. 2 and Tab. 3, respectively. From the results, we can observe that MTRec surpasses MTRec$_n$ in both tasks. Additionally, Bayesian task weighter learner helps improve the performance when MTRec$_n$ does not significantly exceed MTRec$_{rec}$ (e.g., Prec@10 for LastFM in Tab. 2) or MTRec$_{lp}$ (e.g., MRR@10 for LastFM in Tab. 3) due to the improper setting of task weights. This demonstrates the effectiveness of our proposed Bayesian task weight learner in increasing the performance, in addition to its benefit that we do not need to manually set the weights for two tasks in the joint loss.

TABLE 4: Average L2 distance between each meta-path representation and reference representation

Data	Meta-Path	Avg Distance	Meta-Path	Avg Distance
MovieLens	umgm	0.6421	uuum	0.6978
	umum	0.7414	ummm	0.8777
LastFM	uata	0.3555	uuua	0.5871
	uaua	0.6487	uua	0.4478
Yelp	ubcb	0.4121	ubub	0.5789
	ubib	0.3977	uub	0.5566
Yelp-L	ubcb	0.4256	ubub	0.5712
	ubib	0.4001	uub	0.5645

4.2.4 Interpretability of the Recommendation

We now conduct a detailed analysis of MTRec on its interpretability.

First, we investigate the overall impact of different meta-paths in the recommendation results of MTRec. As illustrated in Sec. 3.2.2, the L2 distance between the representation of each meta-path connecting a ground-truth user-item pair and the reference representation $\mathbf{s}^{(item)} - \mathbf{s}^{(user)}$ depicts the importance of each meta-path. Therefore, we calculate the average L2 distance between the embeddings of each meta-path and the corresponding reference for all ground-truth user-item pairs.

From Tab. 4, we can observe that "umgm" and "uata" are most important meta-path for MTRec when it makes a recommendation in MovieLens and LastFM, respectively. For Yelp and Yelp-L, both "ubcb" and "ubib" play a vital role. This observation is consistent with the previous study [6].

We further take two real examples to illustrate how the "*translation*" mechanism used in MTRec can help us understand the recommendations provided by MTRec and enhance the interpretability of RS. Fig. 5 demonstrates the two examples. We examine the recommendation results of MTRec to a user with an anonymous id "joIzw_aUiNvBTuGoytrH7g" in Yelp data set. We call this anonymous user Bob in the sequel.

Example 1 (ubcb). MTRec recommends one bar named "*Tailgaters Sports Bar & Grill*" to Bob, which correctly hits the ground-truth user-item interaction, i.e., Bob has indeed

Fig. 5: Two examples for the interpretability of MTRec

interacted with "*Tailgaters Sports Bar & Grill*". Then, we investigate the L2 distance between relevant meta-path representations and reference representations. We find the meta-path "ubcb" contributes most to the recommendation. The object type "c" (*Category*) in the meta-path "ubcb" gives us the hint. By inspecting into the data, it is found that Bob has visited 21 local businesses in the category of "*Beer, Wine & Spirits*" which explains why "ubcb" leads MTRec to recommend "*Tailgaters Sports Bar & Grill*": Bob frequently goes to lcoal bars. These 21 bars form the first segment (i.e., "ub") of "ubcb". "*Tailgaters Sports Bar & Grill*" and these 21 bars belong to the same category "*Beer, Wine & Spirits*", which provides the third segment (i.e., "cb") and the second segment (i.e., "bc") in ubcb. Finally, "ubcb" leads MTRec to find the correct recommendation target "*Tailgaters Sports Bar & Grill*".

Example 2 (uub). MTRec also correctly recommends one restaurant named "*Oregano's Pizza Bistro*" to Bob. We investigate the L2 distance between relevant meta-path representations and reference representations and find the meta-path "uub" contributes most to this recommendation. We investigate the most similar users of Bob, which indicates that there are edges of "uu" connecting these users to Bob in HIN. We found that the top-4 most similar users have all visited "*Oregano's Pizza Bistro*" before, which means there are edges "ub" connecting these 4 users to the restaurant "*Oregano's Pizza Bistro*". This explains why meta-path "uub" plays a key role in this successful recommendation.

The above two examples show that MTRec is able to provide good interpretabilities for the recommendation results through using the the "*translation*" mechanism introduced in Sec. 3.2.2.

5 Related Work

In this section, we elaborate on three directions of previous works which are related to MTRec.

5.1 Heterogeneous Network Embedding

Heterogeneous network embedding, which studies the embedding problem of HIN through preserving the structural information of meta-path and meta-graph, is one research direction of network embedding. Dong et al. [34] proposed metapath2vec and metapath2vec++, which adopt meta-path based random walks, skip-gram model [35] and heterogeneous negative sampling to learn heterogeneous representations. Fan et al. [36] learns representations based on meta-graph instead of meta-path and proposed metagraph2vec for malware detection. HIN2Vec [27] carries out multiple prediction training tasks jointly to learn latent vectors of objects and meta-paths in HIN. Hussein et al. [37] found that adopting random walks with a jump and stay strategy can better learn the embeddings of HIN and thus selecting and learning on meta-path is not necessary. HERR [38] transcribes the rich and potentially incompatible information from HIN to the embeddings using edge representations. PME [39] utilizes metric learning to capture both first-order and second-order proximities of objects in HIN. Shi et al. [40] introduced the concept of aspects into HIN with each aspect being a unit representing one underlying semantic facet. Instead of preserving information of the network in one semantic space, they proposed ASPEM which encapsulates information regarding each aspect individually. SHNE [41] captures both heterogeneous structural information and unstructured semantic information (e.g., text) of objects. Wang et al. [42] used hyperbolic spaces instead of Euclidean spaces as the proximity measurement for learning HIN embedding. RHINE [43] distinguishes relations into two categories (i.e., affiliation relations and interaction relations) and uses them to model different relations. Wang et al. [44] introduced the attention mechanism into heterogeneous network embedding. The importance between an object and its meta-path based neighborhood and the importance of different meta-paths are learned by object-level attention and semantic-level attention, respectively.

5.2 Traditional Recommendation

Traditional recommender systems (RS) typically rely on collaborative filtering methods (CF), especially matrix factorization (MF) [1], to harness historical user-item interactions for the recommendation. MF models user preferences and item properties by factorizing the user-item interaction matrix into two low-dimensional latent matrices. MF has been successfully deployed in the industry (e.g., Amazon and eBay [1]), due to its effectiveness when handling large-scale data [45]. The cold-start problem, where historical data is not available for new users or items, is one of the most challenging issues in RS. In order to alleviate the cold-start problem, another line of work is to incorporate additional context information, which is also called auxiliary data or side information (e.g., social network [46, 47], review text [48, 49], user grouping data [50, 51], relationships in a graph [52], location [53], image [54] and time-series information [55]), into recommendation models [1].

Recently, the advance of deep learning has fostered the development of RS. Multi-layer perceptron (MLP) [25], Graph Neural Networks (GNN) [29], Variational Autoencoder [30] and other deep neural architectures have been introduced to model the recommendation task and show promising results. Readers can refer to Zhang et al. [56] for a detailed survey.

However, traditional RS only consider homogeneous information and cannot fully model the heterogeneous in-

formation which can be widely observed in real life, i.e., complex objects with different types and rich interactions.

5.3 Recommendations over HIN

Recent decades have witnessed a massive increase of auxiliary data in RS. However, it is difficult to manage and utilize this heterogeneous and complex information using traditional methods for network analysis. HIN provides a flexible way to model the data heterogeneity and characterize the useful structural and semantic information contained in auxiliary data for the recommendation task [7].

Feng and Wang [57] proposed OptRank to utilize heterogeneous information and alleviate the cold-start problem in social tagging systems. Yu et al. [11] measured the similarity of all item pairs along one meta-path and the different preferences on different meta-path semantics were distinguished by linear regression. Then, an unified MF model was used to take advantages of both rating data and meta-path semantics for the recommendation task. Yu et al. [13] introduced the diffusion of user preferences based on similarity matrices defined by different meta-paths and user/item latent features are learned using nonnegative MF on the diffused matrix. With the latent features, they defined a recommendation model and optimized it with Bayesian Ranking. Subsequently, they improved the model by considering personalization [58]. Luo et al. [14] adopted PathSim [8] to measure the relations between users, items and user-item pairs. Finally, they used a unified MF model to incorporate heterogeneous information into social recommendation task. Shi et al. [59] designed SemRec which considers attribute values of links in HIN and personalized weights of different meta-paths for each user in RS. Pham et al. [60] proposed HeteRS, which represents HIN as multiple transition matrices, each of which corresponds to a relation from one to another type of objects. HeteRS transforms the recommendation problem into node proximity calculation problems w.r.t. some query nodes, and then uses the multivariate Markov chain to solve it. Fang et al. [61] harnessed a combination of Bayesian Personalized Ranking model and meta-path based representation learning for music recommendation. FMG [3] adopts MF to learn user/item latent features from user-item similarity matrix, which is obtained from meta-graph. Then FMG feeds the latent features into factorization machine (FM) [62] with Group lasso for the recommendation task.

Previous HIN-based recommendation models mainly rely on meta-path or meta-graph based similarity which suffers from the high computational complexity [16] and may not fully reflect the latent features of users and items in HIN [7]. Recently, there are some attempts to utilize the heterogeneous network embedding in order to mine user and item features in HIN based RS efficiently. Yu et al. [4] proposed to learn user embeddings through meta-path and then compute cosine similarity over user embeddings to identify implicit friends (i.e., users with high similarity) in social RS. NeuACF [31] extracts aspect-level similarity matrices of users and items through different meta-paths and then feeds them into deep neural network with attention to learn aspect-level latent factors in RS. Jiang et al. [63] learned representations for publications from HIN which is constructed from multilingual repositories and then offered recommendations for cross-language citation. Shi et al. [7] proposed HERec, which adopts random walks to generate object sequences and learn the embeddings of objects. Then, the object embeddings are transformed and integrated into an extended MF approach. Chen et al. [5] took advantage of meta-path, neural network, hierarchical attention mechanism and FM to predict ratings in HIN-based recommender. Hu et al. [6] firstly learned the embeddings of meta-path in addition to users and items in HIN-based RS, and considered the interplay between the meta-path and the involved user-item pair in an interaction. They also proposed a co-attention mechanism to mutually improve the representations for meta-path based context, users and items.

6 Conclusion

In this paper, we propose a multi-task learning framework, called MTRec, for recommendation over HIN. MTRec mostly relies on self-attention, to capture the complex semantics in meta-paths. It jointly optimizes tasks of both recommendation and link prediction, and the balance is automatically achieved via using a Bayesian task weight learner. What is more, MTRec provides good interpretabilities of recommendation through a "*translation*" mechanism which is used to model the three-way interactions among users, items and the meta-paths connecting them.

However, MTRec does not achieve a very high precision in some cases (e.g., around 0.2 for Prec@10 in Yelp), though it already exceeds existing methods. This observation indicates that HIN-based recommendation problem is far from being addressed completely. Moreover, MTRec does not *explicitly* leverage users' new coming interactions and perform incremental learning (i.e., data sample-level learning). The bidirectional information flow between HIN and RS in current MTRec is *implicit*, i.e., the parameters of two components are updated according to the information flow between them (i.e., feature-level learning). In the future, we plan to study if the precision of MTRec can be further improved or not, by considering data sample-level information flow.

7 Acknowledgment

This work was supported by the National Natural Science Foundation of China (no. 61972328).

References

[1] Y. Shi, M. Larson, and A. Hanjalic, "Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges," *ACM Comput. Surv.*, vol. 47, no. 1, pp. 3:1–3:45, 2014.

[2] C. Shi, Y. Li, J. Zhang, Y. Sun, and P. S. Yu, "A survey of heterogeneous information network analysis," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 1, pp. 17–37, 2017.

[3] H. Zhao, Q. Yao, J. Li, Y. Song, and D. L. Lee, "Meta-graph based recommendation fusion over heterogeneous information networks," in *KDD*, 2017, pp. 635–644.

[4] J. Yu, M. Gao, J. Li, H. Yin, and H. Liu, "Adaptive implicit friends identification over heterogeneous network for social recommendation," in *CIKM*, 2018, pp. 357–366.

[5] L. Chen, Y. Liu, Z. Zheng, and P. S. Yu, "Heterogeneous neural attentive factorization machine for rating prediction," in *CIKM*, 2018, pp. 833–842.

[6] B. Hu, C. Shi, W. X. Zhao, and P. S. Yu, "Leveraging meta-path based context for top-N recommendation with A neural co-attention model," in *KDD*, 2018, pp. 1531–1540.

[7] C. Shi, B. Hu, W. X. Zhao, and P. S. Yu, "Heterogeneous information network embedding for recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 2, pp. 357–370, 2019.

[8] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu, "Pathsim: Meta path-based top-k similarity search in heterogeneous information networks," *PVLDB*, vol. 4, no. 11, pp. 992–1003, 2011.

[9] Y. Fang, W. Lin, V. W. Zheng, M. Wu, K. C. Chang, and X. Li, "Semantic proximity search on graphs with metagraph-based learning," in *ICDE*, 2016, pp. 277–288.

[10] Z. Huang, Y. Zheng, R. Cheng, Y. Sun, N. Mamoulis, and X. Li, "Meta structure: Computing relevance in large heterogeneous information networks," in *KDD*, 2016, pp. 1595–1604.

[11] X. Yu, X. Ren, Q. Gu, Y. Sun, and J. Han, "Collaborative filtering with entity similarity regularization in heterogeneous information networks," in *IJCAI HINA*, 2013.

[12] Y. Zhang and X. Chen, "Explainable recommendation: A survey and new perspectives," *arXiv Preprint*, 2018. [Online]. Available: https://arxiv.org/abs/1804.11192

[13] X. Yu, X. Ren, Y. Sun, B. Sturt, U. Khandelwal, Q. Gu, B. Norick, and J. Han, "Recommendation in heterogeneous information networks with implicit user feedback," in *RecSys*, 2013, pp. 347–350.

[14] C. Luo, W. Pang, Z. Wang, and C. Lin, "Hete-cf: Social-based collaborative filtering recommendation using heterogeneous relations," in *ICDM*, 2014, pp. 917–922.

[15] H. Zhao, Q. Yao, Y. Song, J. T. Kwok, and D. L. Lee, "Learning with heterogeneous side information fusion for recommender systems," *arXiv Preprint*, 2018. [Online]. Available: https://arxiv.org/abs/1801.02411

[16] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *IEEE Trans. Knowl. Data Eng.*, 2018.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*, 2017, pp. 6000–6010.

[18] A. Galassi, M. Lippi, and P. Torroni, "Attention, please! A critical review of neural attention models in natural language processing," *arXiv Preprint*, 2019. [Online]. Available: https://arxiv.org/abs/1902.02181

[19] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 12, pp. 2724–2743, 2017.

[20] Y. Liu, H. Li, A. García-Durán, M. Niepert, D. Oñoro-Rubio, and D. S. Rosenblum, "MMKG: multi-modal knowledge graphs," in *ESWC*, vol. 11503, 2019, pp. 459–474.

[21] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *NIPS*, 2013, pp. 2787–2795.

[22] W. Kang and J. J. McAuley, "Self-attentive sequential recommendation," in *ICDM*, 2018, pp. 197–206.

[23] S. Ruder, "An overview of multi-task learning in deep neural networks," *arXiv Preprint*, 2017. [Online]. Available: https://arxiv.org/abs/1706.05098

[24] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *CVPR*, 2018, pp. 7482–7491.

[25] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua, "Neural collaborative filtering," in *WWW*, 2017, pp. 173–182.

[26] H. Yu, C. Hsieh, S. Si, and I. S. Dhillon, "Scalable coordinate descent approaches to parallel matrix factorization for recommender systems," in *ICDM*, 2012, pp. 765–774.

[27] T. Fu, W. Lee, and Z. Lei, "Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning," in *CIKM*, 2017, pp. 1797–1806.

[28] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: bayesian personalized ranking from implicit feedback," in *UAI*, 2009, pp. 452–461.

[29] X. Wang, X. He, M. Wang, F. Feng, and T. Chua, "Neural graph collaborative filtering," in *SIGIR*, 2019, pp. 165–174.

[30] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, "Variational autoencoders for collaborative filtering," in *WWW*, 2018, pp. 689–698.

[31] X. Han, C. Shi, S. Wang, P. S. Yu, and L. Song, "Aspect-level deep collaborative filtering via heterogeneous information networks," in *IJCAI*, 2018, pp. 3393–3399.

[32] M. Quadrana, P. Cremonesi, and D. Jannach, "Sequence-aware recommender systems," *ACM Comput. Surv.*, vol. 51, no. 4, pp. 66:1–66:36, 2018.

[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[34] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *KDD*, 2017, pp. 135–144.

[35] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013, pp. 3111–3119.

[36] Y. Fan, S. Hou, Y. Zhang, Y. Ye, and M. Abdulhayoglu, "Gotcha - sly malware!: Scorpion A metagraph2vec based malware detection system," in *KDD*, 2018, pp. 253–262.

[37] R. Hussein, D. Yang, and P. Cudré-Mauroux, "Are meta-paths necessary?: Revisiting heterogeneous graph embeddings," in *CIKM*, 2018, pp. 437–446.

[38] Y. Shi, Q. Zhu, F. Guo, C. Zhang, and J. Han, "Easing embedding learning by comprehensive transcription of heterogeneous information networks," in *KDD*, 2018, pp. 2190–2199.

[39] H. Chen, H. Yin, W. Wang, H. Wang, Q. V. H. Nguyen, and X. Li, "PME: projected metric embedding on heterogeneous networks for link prediction," in *KDD*,

2018, pp. 1177–1186.
[40] Y. Shi, H. Gui, Q. Zhu, L. M. Kaplan, and J. Han, "Aspem: Embedding learning by aspects in heterogeneous information networks," in *SDM*, 2018, pp. 144–152.
[41] C. Zhang, A. Swami, and N. V. Chawla, "SHNE: representation learning for semantic-associated heterogeneous networks," in *WSDM*, 2019, pp. 690–698.
[42] X. Wang, Y. Zhang, and C. Shi, "Hyperbolic heterogeneous information network embedding," in *AAAI*, 2019, pp. 5337–5344.
[43] Y. Lu, C. Shi, L. Hu, and Z. Liu, "Relation structure-aware heterogeneous information network embedding," in *AAAI*, 2019, pp. 4456–4463.
[44] X. Wang, H. Ji, C. Shi, B. Wang, P. Cui, P. Yu, and Y. Ye, "Heterogeneous graph attention network," in *WWW*, 2019.
[45] H. Li, T. N. Chan, M. L. Yiu, and N. Mamoulis, "FEXIPRO: fast and exact inner product retrieval in recommender systems," in *SIGMOD Conference*, 2017, pp. 835–850.
[46] H. Li, D. Wu, W. Tang, and N. Mamoulis, "Overlapping community regularization for rating prediction in social recommender systems," in *RecSys*, 2015, pp. 27–34.
[47] H. Li, D. Wu, and N. Mamoulis, "A revisit to social network-based recommender systems," in *SIGIR*, 2014, pp. 1239–1242.
[48] C. Wang, M. Niepert, and H. Li, "LRMM: learning to recommend with missing modalities," in *EMNLP*, 2018, pp. 3360–3370.
[49] A. García-Durán, R. Gonzalez, D. Oñoro-Rubio, M. Niepert, and H. Li, "Transrev: Modeling reviews as translations from users to items," *arXiv Preprint*, 2018. [Online]. Available: https://arxiv.org/abs/1801.10095
[50] D. Ding, H. Li, Z. Huang, and N. Mamoulis, "Efficient fault-tolerant group recommendation using alpha-beta-core," in *CIKM*, 2017, pp. 2047–2050.
[51] H. Li, Y. Liu, Y. Qian, N. Mamoulis, W. Tu, and D. W. Cheung, "HHMF: hidden hierarchical matrix factorization for recommender systems," *Data Min. Knowl. Discov.*, vol. 33, no. 6, pp. 1548–1582, 2019.
[52] Y. Qian, H. Li, N. Mamoulis, Y. Liu, and D. W. Cheung, "Reverse k-ranks queries on large graphs," in *EDBT*, 2017, pp. 37–48.
[53] Z. Lu, H. Li, N. Mamoulis, and D. W. Cheung, "HBGG: a hierarchical bayesian geographical model for group recommendation," in *SDM*, 2017, pp. 372–380.
[54] C. Wang, M. Niepert, and H. Li, "Recsys-dan: Discriminative adversarial networks for cross-domain recommender systems," *IEEE Trans. Neural Netw. Learning Syst.*, 2019.
[55] H. Li, Y. Liu, N. Mamoulis, and D. S. Rosenblum, "Translation-based sequential recommendation for complex users on sparse data," *IEEE Trans. Knowl. Data Eng.*, 2019.
[56] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Comput. Surv.*, vol. 52, no. 1, pp. 5:1–5:38, 2019.
[57] W. Feng and J. Wang, "Incorporating heterogeneous information for personalized tag recommendation in social tagging systems," in *KDD*, 2012, pp. 1276–1284.
[58] X. Yu, X. Ren, Y. Sun, Q. Gu, B. Sturt, U. Khandelwal, B. Norick, and J. Han, "Personalized entity recommendation: a heterogeneous information network approach," in *WSDM*, 2014, pp. 283–292.
[59] C. Shi, Z. Zhang, P. Luo, P. S. Yu, Y. Yue, and B. Wu, "Semantic path based personalized recommendation on weighted heterogeneous information networks," in *CIKM*, 2015, pp. 453–462.
[60] T. N. Pham, X. Li, G. Cong, and Z. Zhang, "A general recommendation model for heterogeneous networks," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 12, pp. 3140–3153, 2016.
[61] Q. Fang, L. Liu, J. Yu, and J. Wen, "Meta-path based heterogeneous graph embedding for music recommendation," in *ICONIP (3)*, vol. 11303, 2018, pp. 101–113.
[62] S. Rendle, "Factorization machines," in *ICDM*, 2010, pp. 995–1000.
[63] Z. Jiang, Y. Yin, L. Gao, Y. Lu, and X. Liu, "Cross-language citation recommendation via hierarchical representation learning on heterogeneous graph," in *SIGIR*, 2018, pp. 635–644.

Hui Li is currently an assistant professor in the School of Informatics, Xiamen University. His research interests include data mining and data management with applications in recommender systems and knowledge graph. He received his B.Eng. degree in Software Engineering from Central South University (2012), and his MPhil and PhD degrees in Computer Science from University of Hong Kong (2015, 2018).

Yanlin Wang is a researcher in Microsoft Research Asia. Her research interests include programming languages and data engineering, and the applications of their intersection in the big data area. She obtained her B.Eng. degree in Computer Science and Technology from Zhejiang University in 2014 and her PhD degree in Computer Science from University of Hong Kong in 2019.

Ziyu Lyu is currently an assistant professor in Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences. She received a Ph.D. degree in Computer Science from the University of Hong Kong in 2016. Her work includes research into spatio-temporal databases, recommender systems, text mining, and machine learning.

Jieming Shi received his PhD in computer science from University of Hong Kong. He is now a Research Fellow at National University of Singapore. His research interests include large scale graph algorithms, recommendations, and knowledge graphs.