World Wide Web (2021) 24:1411–1437
https://doi.org/10.1007/s11280-021-00910-6

GSIRec: Learning with graph side information for recommendation

Anchen Li[1] · **Bo Yang**[1]

Received: 8 May 2020 / Revised: 16 May 2021 / Accepted: 10 June 2021 /
Published online: 5 July 2021

Abstract

Collaborative filtering (CF) is one of the dominant techniques used in modern recommender systems. Traditional CF-based methods suffer from issues of data sparsity and cold start. Therefore, side information has been widely utilized by researchers to address these problems. Most side information is typically heterogeneous and in the form of the graph structure. In this work, we propose a deep end-to-end recommendation framework named GSIRec to make full use of the graph side information. Specifically, GSIRec derives a multi-task learning approach that introduces a side information task to assist the recommendation task. The key idea is that we design a delicate *knowledge assistance module* to be the bridge between tasks, which captures useful knowledge to complement each task. Also, we utilize a graph attention method to exploit the topological structure of side information to enhance recommendation. To show the wide application and flexibility of our framework, we integrate side information from two aspects: social networks (for users) and knowledge graphs (for items). We apply GSIRec in two recommendation scenarios: social-aware recommendation and knowledge-aware recommendation. To evaluate the effectiveness of our framework, we conduct extensive experiments with four real-world public datasets. The results reveal that GSIRec consistently outperforms the state-of-the-art methods on the rating prediction task and top-K recommendation task. Moreover, GSIRec can alleviate data sparsity and cold start issues to some extent.

Keywords Side information recommendation · Multi-task learning · Graph neural network · Social network · Knowledge graph

✉ Bo Yang
ybo@jlu.edu.cn

Anchen Li
liac18@mails.jlu.edu.cn

[1] College of Computer Science and Technology, Jilin University, Changchun, China

1 Introduction

In the era of information explosion, recommender systems have been playing an indispensable role in meeting user preferences by recommending relevant items [22, 46, 51, 69, 74]. Collaborative filtering (CF) [21] is one of the most widely used techniques, which usually projects users and items in a low-dimensional vector space for predicting users' preferences for items. Since traditional CF-based methods rely heavily on user-item interaction data to learn ID-based embeddings, they are often impeded by data sparsity and cold start problems. To mitigate these drawbacks, side information is widely used by a large body of research. Users and items are always associated with various side information. In this work, we mainly consider two typical kinds of side information with graph structure: social networks (for users) and knowledge graphs (for items). We focus on two recommendation scenarios: social-aware recommendation and knowledge-aware recommendation.

Social-aware recommendation Social networks contain social relationships (e.g. friendship or trust) between users. According to the social influence theory [1, 3, 31], users' preferences are influenced by their neighbors in the social network. For example, in Figure 1(left), shopping may become *User A*'s interest, since most of her friends are shopping fans. Social-aware recommendation aims at harnessing social networks to improve the performance of recommendation. Previous research has been attempted to use matrix factorization [68, 73, 78], regularization [39], and trust propagation [40] to learn the preference of users. As deep learning has grown in prominence, researchers merged social networks into recommendation by using different deep learning technologies including multi-layer perceptron [14], autoencoder [43], graph neural networks [15, 67].

Knowledge-aware recommendation Knowledge graphs provide rich facts and relations about items. Exploiting the semantic information from knowledge graphs is particularly

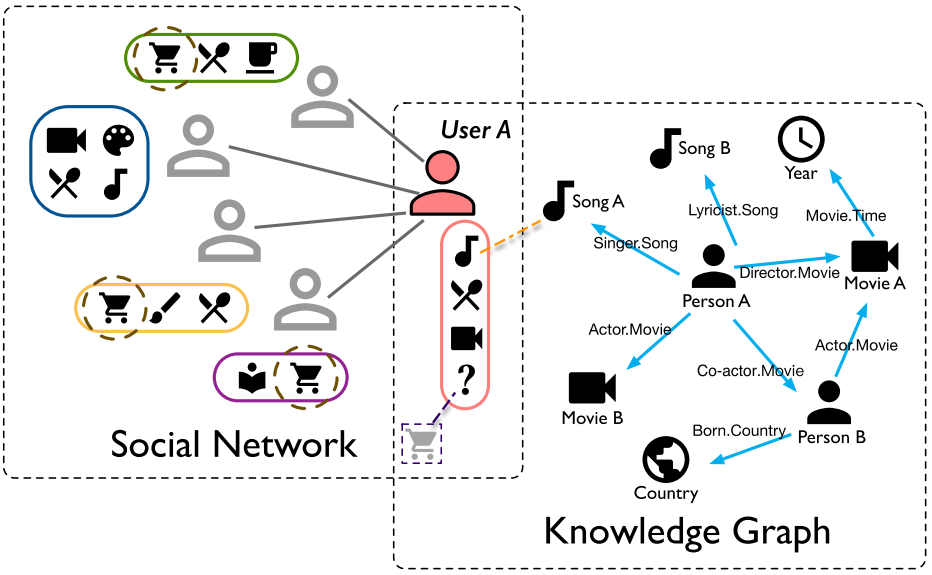


Figure 1 Social network (left) and knowledge graph (right)

helpful to understand users and items. For instance, in Figure 1(right), *User A* is fond of music, and one of her favorite songs is linked with its namesake *Song A* in the knowledge graph. The singer *Person A* of *Song A* is also the lyricist of *Song B*. In this case, it would make sense that *User A* would like to listen to *Song B*. We also find *Person A* is the director and actor of *Movie A* and *Movie B*, and hence *User A* may love these two movies due to her preference for movies. Knowledge-aware recommendation integrates knowledge graphs into recommender systems and has attracted considerable interest. Some researchers derive different meta-paths [72] or meta-graphs [77] to mine semantic information from knowledge graphs. Another line of work [62, 76] uses knowledge graph embedding methods to integrate knowledge graphs into recommendation. Recently, researchers use propagation-based methods [55, 65] to consider semantic information and connectivity information in the whole knowledge graph.

Our approach Nevertheless, we argue that most methods lack techniques to take full advantage of the graph side information. Most methods utilize the same user (item) representation in both recommendation and side information domains, which is likely to restrict user (item) representation learning in their respective domain and may further cause a negative knowledge transfer from the side information. To address the aforementioned shortcoming, for each user, we employ two kinds of embedding vectors to encode representations of users in the recommendation task and social network embedding task in social-aware recommendation. Similar to the user, item representations are also mapped into two low-dimensional vector spaces in the recommendation task and knowledge graph embedding task in knowledge-aware recommendation.

In this paper, we propose our multi-task learning framework GSIRec that combines the recommendation task with a side information task to enhance the recommendation performance. Specifically, GSIRec introduces the social network embedding task and knowledge graph embedding task for social-aware recommendation and knowledge-aware recommendation, respectively. The users are connected in the social network and the items they love may appear as corresponding entities in the knowledge graph, hence side information tasks have great relevance to the recommendation task. To transfer the knowledge between the recommendation task and side information tasks, we design a special *knowledge assistance module* as the bridge between these tasks. Such a module captures useful knowledge from latent representations in related tasks and integrates the knowledge to assist each task in improving its performance. There are two advantages of utilizing multi-task learning: (i) leveraging the training signals of related tasks can enhance performance for each task; and (ii) learning related tasks at a time reduces the risk of overfitting compared with single-task learning [45]. To further exploit the topological structure of the side information, we use graph attention network in the recommendation task. We design an attention mechanism to consider node information of the user (item) neighbors in graphs. The recommendation task and side information tasks are optimized alternatively, which enables the whole framework to be more flexible.

Contribution and organization In summary, our contributions in this paper are as follows:

- We proposed a deep end-to-end framework GSIRec for graph side information recommendation.
- We provide a knowledge assistance module that can fully transfer knowledge and capture the interplay between related tasks to assist each task in enhancing its performance.

– Extensive experiments demonstrate that GSIRec consistently outperforms the state-of-the-art recommendation solutions in different recommendation scenarios. Also, GSIRec shows advantages in mitigating the data sparsity and cold start issues.

The remainder of this paper is organized as follows. Our framework GSIRec is described in Section 2. In Section 3, we conduct experiments on four real-world datasets and present the experimental results. In Section 4, we review work related to our method GSIRec, followed by conclusions and future work in Section 5.

2 Methodology

In this section, we firstly introduce key mathematical notations and formulate the problems, then describe our GSIRec framework and define each component in more depth.

2.1 Notations and problem formulation

Table 1 depicts key notations in this paper. In a recommendation scenario, we suppose there are N users $U = \{u_1, u_2, \ldots, u_N\}$ and M items $V = \{v_1, v_2, \ldots, v_M\}$. We define $\mathbf{R} \in \mathbb{R}^{N \times M}$ as the user-item interaction matrix whose element r_{io} indicates the rating score or implicit feedback (0 or 1) from u_i to v_o. In addition, we consider social networks and knowledge graphs as two kinds of side information. We define the matrix $\mathbf{S} \in \mathbb{R}^{N \times N}$ as the social network among users U, where its element $s_{ij} = 1$ if u_i trusts u_j and zero otherwise. As for the knowledge graph, it consists of real-world entities $\mathcal{E} = \{e_1, e_2, \ldots, e_W\}$ and relations $\mathcal{R} = \{r_1, r_2, \ldots, r_C\}$ among these entities. Since an item $v \in V$ is likely to appear as an entity $e \in \mathcal{E}$ in the knowledge graph $\mathbf{G}$, entity set $\mathcal{E}$ consists of items $V (V \subseteq \mathcal{E})$ and non-items $\mathcal{E} \setminus V$. Formally, we define the knowledge graph as $\mathbf{G} = \left\{ \left(e_o, r_h, e_p\right) | e_o, e_p \in \mathcal{E},\ r_h \in \mathcal{R} \right\}$ whose the i-th element $\left(e_o, r_h, e_p\right)_i$ is a triple indicating that there is a relation r_h between entity e_o and entity e_p. For user u_i, $\mathbf{u}_i^r \in \mathbb{R}^d$ and $\mathbf{u}_i^s \in \mathbb{R}^d$ denote user embeddings corresponding to the recommendation domain and social domain respectively, where d is the embedding size. For item v_o (corresponding to entity e_o), we utilize $\mathbf{e}_o^r \in \mathbb{R}^d$ and $\mathbf{e}_o^k \in \mathbb{R}^d$ to denote its embeddings corresponding to the recommendation domain and knowledge graph domain respectively. In addition, we use $\mathbf{r}_h \in \mathbb{R}^d$ to denote embedding for relation r_h.

The goal of GSIRec is to use the user-item interaction matrix $\mathbf{R}$ and the side information to predict user's personalized interests in items. Specifically, for social-aware recommendation, given the user-item interaction information $\mathbf{R}$ and social network $\mathbf{S}$, GSIRec aims to learn a prediction function $\hat{r}_{uv} = \mathcal{F}(u, v|\Theta, \mathbf{R}, \mathbf{S})$, where $\hat{r}_{uv}$ is the predicted rating or preference probability from user u to item v, and Θ is the framework parameters of function $\mathcal{F}$. Similarly, for knowledge-aware recommendation, given the user-item interaction information $\mathbf{R}$ and knowledge graphs $\mathbf{G}$, GSIRec aims to learn a prediction function $\hat{r}_{uv} = \mathcal{F}(u, v|\Theta, \mathbf{R}, \mathbf{G})$.

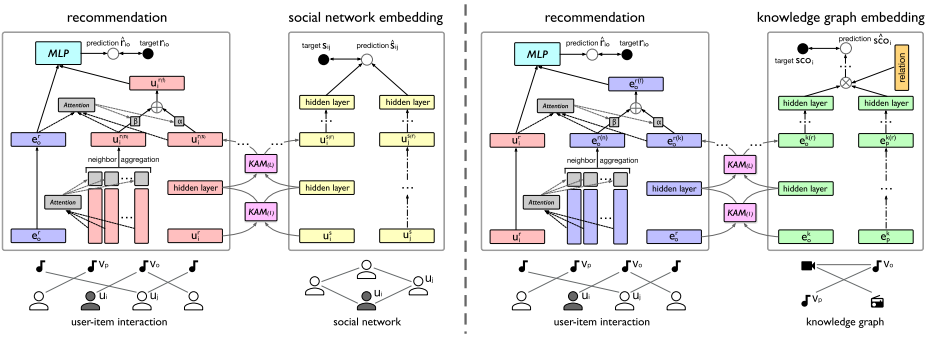
Figure 2 shows the overview of the architecture of GSIRec in two recommendation scenarios: social-aware recommendation (left) and knowledge-aware recommendation (right). The whole framework consists of two tasks: recommendation task and side information task (social network embedding or knowledge graph embedding). Two tasks are bridged by the knowledge assistance modules. In what follows, we introduce the knowledge assistance module, then apply GSIRec in social-aware recommendation and knowledge-aware recommendation, finally discuss the learning algorithm for GSIRec.

Table 1 Key notations

Symbols	Definitions and descriptions
U	Set of users
V	Set of items
$\mathbf{R}$	User-item interaction matrix
$\mathbf{S}$	User-user social link matrix
$\mathbf{G}$	Knowledge graph
$\mathcal{E}$	Set of entities in knowledge graph
$\mathcal{R}$	Set of relations in knowledge graph
$\mathbf{u}_i^r$	Embedding of user u_i in recommendation domain
$\mathbf{u}_i^s$	Embedding of user u_i in social domain
$\mathbf{e}_o^r$	Embedding of item v_o in recommendation domain
$\mathbf{e}_o^k$	Embedding of item v_o in knowledge graph domain
$\mathbf{r}_h$	Embedding of relation r_h
$\hat{r_{io}}$	Predicted rating from u_i to v_o
$\hat{s_{ij}}$	Predicted trust relation probability between u_i and u_j
$\hat{sco_i}$	Predicted score of triple $\left(e_o, r_h, e_p\right)_i$

2.2 Knowledge assistance module

To obtain communication and interaction between recommendation task and social network embedding task (or recommendation task and knowledge graph embedding task), we design a delicate *knowledge assistance module*, as shown in Figure 2 (the pink blocks). The design of the knowledge assistance module is motivated by feature sharing in multi-task learning [24, 70]. Specifically, the knowledge assistance module takes two user embeddings $\mathbf{u}_i^r$ and $\mathbf{u}_i^s$ of the user u_i as inputs (or takes the two entity embeddings $\mathbf{e}_o^r$ and $\mathbf{e}_o^k$ of an item v_o as inputs), enables them to capture useful knowledge from each other, and update themselves. Note that we set the knowledge assistance modules in the low-level layers in both recommendation task and side information task, because sharing of high-level feature layers may result in negative transfer problems in multi-task learning according to recent research [36,



(a) Social-aware recommendation (b) Knowledge-aware recommendation

Figure 2 A schematic view of GSIRec for social-aware recommendation (left) and knowledge-aware recommendation (right). KAM and MLP indicate the knowledge assistance module and multi-layer perceptron

64, 70]. We devise two kinds of knowledge assistance modules: attention-based (left) and gate-based (right), which are shown in Figure 3. Without loss of generality, we denote x_1, x_2 as two inputs and y_1, y_2 as two outputs of a knowledge assistance module:

$$y_1 = \mathcal{K}(x_1, x_2)\langle x_1 \rangle, \tag{1}$$

$$y_2 = \mathcal{K}(x_1, x_2)\langle x_2 \rangle, \tag{2}$$

where $\mathcal{K}$ is the knowledge assistance module and we use a suffix $\langle x_1 \rangle$ or $\langle x_2 \rangle$ to obtain one of its outputs. We will elaborate on these two kinds of knowledge assistance modules in the following.

2.2.1 Attention-based knowledge assistance module

Inspired by [54], we first utilize a key matrix $M_k \in \mathbb{R}^{d\times d}$ and a value matrix $M_v \in \mathbb{R}^{d\times d}$ to map input vectors x_1, x_2, and then match them as follows:

$$x_1^{m1} = M_k x_1 \odot M_v x_1, \quad x_1^{m2} = M_k x_1 \odot M_v x_2, \tag{3}$$

$$x_2^{m1} = M_k x_2 \odot M_v x_1, \quad x_2^{m2} = M_k x_2 \odot M_v x_2, \tag{4}$$

where $\odot$ is the element-wise product of vectors.

Then we use an attention mechanism to obain the final output representations y_1 and y_2:

$$y_1 = \alpha_1 x_1^{m1} + \beta_1 x_1^{m2}, \tag{5}$$

$$y_2 = \alpha_2 x_2^{m1} + \beta_2 x_2^{m2}, \tag{6}$$

where $\alpha_.$ and $\beta_.$ are the attention weights which characterize the importance of corresponding to the vectors. Specifically, $\alpha_.$ and $\beta_.$ are defined as:

$$\alpha_1 = \frac{exp(dot(w_1, x_1^{m1}))}{exp(dot(w_1, x_1^{m1})) + exp(dot(w_1, x_1^{m2}))} = 1 - \beta_1, \tag{7}$$

$$\alpha_2 = \frac{exp(dot(w_2, x_2^{m1}))}{exp(dot(w_2, x_2^{m1})) + exp(dot(w_2, x_2^{m2}))} = 1 - \beta_2, \tag{8}$$

where $w_. \in \mathbb{R}^d$ is the parameter of the attention mechanisms and dot denotes the inner product of two vectors.

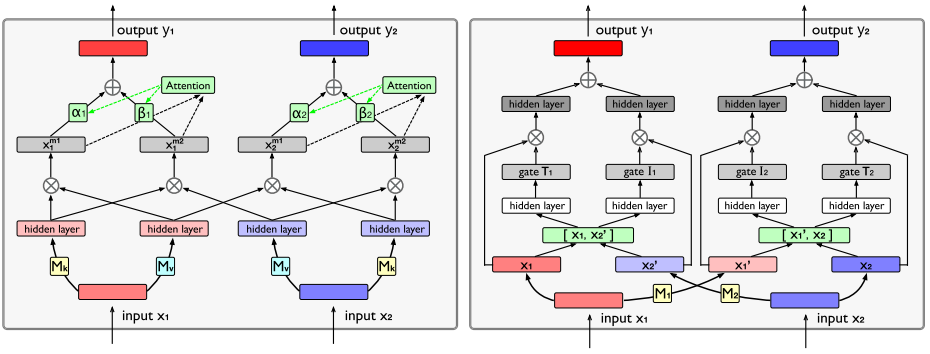


Figure 3 Attention-based knowledge assistance module (left) and gate-based knowledge assistance module (right)

2.2.2 Gate-based knowledge assistance module

First, we utilize two trainable weight matrices $M_1 \in \mathbb{R}^{d\times d}$, $M_2 \in \mathbb{R}^{d\times d}$ to map input vectors x_1, x_2 into new spaces:

$$x_2' = M_1 x_2, \;\; x_1' = M_2 x_1. \tag{9}$$

Then we perform concatenation operations to get two fusion vectors F_1 and F_2 as follows:

$$F_1 = \left[x_1, \; x_2'\right], \;\; F_2 = \left[x_1', \; x_2\right]. \tag{10}$$

Next F_1 and F_2 pass through two non-linear transformations and get two gated structures: input gate I_i and transform gate T_i:

$$I_1 = \sigma\left(w_{1i} F_1 + b_{1i}\right), \; T_1 = \sigma\left(w_{1t} F_1 + b_{1t}\right), \tag{11}$$

$$I_2 = \sigma\left(w_{2i} F_2 + b_{2i}\right), \; T_2 = \sigma\left(w_{2t} F_2 + b_{2t}\right), \tag{12}$$

where $w_{\cdot\cdot} \in \mathbb{R}^{2d\times d}$, $b_{\cdot\cdot} \in \mathbb{R}^d$ are the weight and bias parameters, and σ denotes the sigmoid function $\sigma(x) = 1/\left(1 + e^{-x}\right)$. With these gates structures, we will get the final outputs y_1 and y_2:

$$y_1 = I_1 \odot x_2' + T_1 \odot x_1, \tag{13}$$

$$y_2 = I_2 \odot x_1' + T_2 \odot x_2. \tag{14}$$

2.2.3 Comparing two knowledge assistance modules

We compare our two knowledge assistance modules from four aspects, including expressive power, output space, the number of parameters, and computational complexity.

Expressive power For the attention-based knowledge assistance module, we explicitly utilize the attention mechanisms to calculate the transfer weights. Specifically, we utilize learnable matrices M_k and M_v to make enough interaction and communication between input vectors, and then utilize the attention mechanisms to calculate the transfer weights $\alpha_\cdot$, $\beta_\cdot$ between feature vectors. As for the gate-based knowledge assistance module, we introduce gated structures $I_\cdot$, $T_\cdot$ to control the error flow from related tasks and reduce the risk of negative transfer in multi-task learning to a certain extent. Both knowledge assistance modules utilize learnable weight matrices for more fine-grained knowledge transfer between tasks. In the experiments section, we will further evaluate the effectiveness of these two knowledge assistance modules.

Output space Both gate-based knowledge assistance module and attention-based knowledge assistance module take two embeddings as inputs (i.e., one of them belongs to the recommendation task, and the other belongs to the side information task), enables these two embeddings to capture useful knowledge from each other and update themselves. After the update, both gate-based knowledge assistance module and attention-based knowledge assistance module output the updated embeddings. The updated embeddings will send back the corresponding task. Through knowledge assistance modules, representations from related tasks can complement each other, assisting related tasks in improving generalization.

Space complexity For an attention-based knowledge assistance module, we introduce two learnable matrices $M_k \in \mathbb{R}^{d\times d}$, $M_v \in \mathbb{R}^{d\times d}$, and attention weights $w_1 \in \mathbb{R}^d$, $w_2 \in \mathbb{R}^d$. Therefore, the number of parameters of an attention-based knowledge assistance module

is $\theta_a = \{M_k, M_v, w_1, w_2\}$. For a gate-based knowledge assistance module, we introduce two learnable matrices $M_1 \in \mathbb{R}^{d\times d}$ and $M_2 \in \mathbb{R}^{d\times d}$, and gated structure weights $w_{1i} \in \mathbb{R}^{2d\times d}$, $w_{1t} \in \mathbb{R}^{2d\times d}$, $w_{2i} \in \mathbb{R}^{2d\times d}$, $w_{2t} \in \mathbb{R}^{2d\times d}$, $b_{1i} \in \mathbb{R}^d$, $b_{1t} \in \mathbb{R}^d$, $b_2 i \in \mathbb{R}^d$, $b_2 t \in \mathbb{R}^d$. Therefore, the number of parameters of a gate-based knowledge assistance module is $\theta_g = \{M_1, M_2, w_{1i}, w_{1t}, w_{2i}, w_{2t}, b_{1i}, b_{1t}, b_{2i}, b_{2t}\}$. Compare the number of parameters of two knowledge assistance modules, gate-based knowledge assistance module introduces more parameters.

Time complexity For an attention-based knowledge assistance module, it takes two embeddings $x_1 \in \mathbb{R}^d, x_2 \in \mathbb{R}^d$ as inputs, the time consumption in mapping and matching process is $O(8\times d^2+4\times d)$. The time consumption in the attention mechanism is $O(4\times d)$. Therefore, the time complexity of a attention-based knowledge assistance module is $O(8\times d^2 + 8\times d) \Rightarrow O(d^2)$. For a gate-based knowledge assistance module, it takes two embeddings $x_1 \in \mathbb{R}^d, x_2 \in \mathbb{R}^d$ as inputs, the time consumption in mapping process is $O(2\times d^2)$. The time consumption in gated structures is $O(8\times d^2+4\times d)$. Therefore, the space complexity of a gate-based knowledge assistance module is $O(10\times d^2+4\times d) \Rightarrow O(d^2)$. We can find that the two knowledge assistance modules have similar time complexity.

2.2.4 Advantage

Recently, many recommenders leverage multi-task learning and design different transfer units to enhance recommendation performance, such as the cross-connection unit [24] and the cross-compress unit [64]. These transfer units and our method GSIRec follow the parameter sharing paradigm for multi-task Learning [45]. However, our methodology is different from theirs. Compared with the above transfer units, GSIRec uses attention mechanisms and gated structures to design completely different but more effective transfer units for knowledge transfer between related tasks.

The advantages of the knowledge assistance module are two-fold: (i) attention mechanism and gated structures in the knowledge assistance module can capture and integrate useful information to update representations in a more fine-grained way; and (ii) enough interaction and communication between embedding vectors enable related tasks to complement each other and enhance their performance instead of improving one task at the expense of poor performance on other tasks. We will further compare our knowledge assistance modules with different transfer units in the experiments.

2.3 Social-aware recommendation

In social-aware recommendation, we introduce a social network embedding task to assist the recommendation task.

2.3.1 Recommendation task

Given user u_i and item v_o, such task predicts rating or preference probability $\hat{r}_{io}$ from user u_i to item v_o. We will learn two user embeddings from different perspectives. We first utilize L knowledge assistance modules to obtain social enhanced user embedding $\mathbf{u}_i^{r(s)}$ as follows:

$$\mathbf{u}_i^{r(s)} = \mathcal{K}_L\left(\mathbf{u}_i^r, \mathbf{u}_i^s\right)\left\langle \mathbf{u}_i^r \right\rangle. \tag{15}$$

In addition, we consider neighbors of user u_i to further exploit the topological structure of the social network. A common way to aggregate neighbor information is the mean pool-

ing operation. However, such an operation ignores different weights of neighbors. In our paper, we utilize the attention mechanism to study the importance of different neighbors in the graph for better neighbor aggregation. Specifically, we aggregate the user u_i's neighbor information by linear combination as follows:

$$\mathbf{u}_i^{nei} = \sum_{u_j \in nei(u_i)} att(u_i, u_j)\mathbf{u}_j^r, \tag{16}$$

where $nei(u_i)$ is the u_i's neighbor set. att is function to study the importance of different neighbors for the given user, which is defined as follows:

$$att(u_i, u_j) = \frac{exp\left(dot\left(\mathbf{u}_i^r, \mathbf{u}_j^r\right)\right)}{\sum_{u_k \in nei(u_i)} exp\left(dot\left(\mathbf{u}_i^r, \mathbf{u}_k^r\right)\right)}. \tag{17}$$

In social network, the size of $nei(\cdot)$ is different for different users. Following the approaches in [19, 65], we sample a fixed-size set $nei^s(u)$ for each user's neighbors to keep training the whole framework more efficient, where $|nei^s(u)| = n_s$. Such sampling strategy can facilitate computation on large graphs. In this way, we compute the neighbor enhanced embedding $\mathbf{u}_i^{r(n)}$ for user u_i by replacing $nei(\cdot)$ with $nei^s(\cdot)$:

$$\mathbf{u}_i^{r(n)} = \mathcal{A}\left(w\left(\mathbf{u}_i^r + \mathbf{u}_i^{nei^s}\right)\right), \tag{18}$$

where $w \in \mathbb{R}^{d \times d}$ is the weight and $\mathcal{A}$ is the activation function.

Through a single neighbor aggregation operation, neighbor enhanced embedding $\mathbf{u}_i^{r(n)}$ is dependent on itself as well as the direct social neighbors. We can further extend neighbor aggregation operation to multi-hop neighbors. More formally, to obtain J-order neighbor information (J is a pre-defined value), neighbor enhanced embedding $\mathbf{u}_{i(J)}^{r(n)}$ is defined as:

$$\mathbf{u}_{i(J)}^{r(n)} = \mathcal{A}\left(w_{(J-1)}\left(\mathbf{u}_{i(J-1)}^r + \mathbf{u}_{i(J-1)}^{nei^s}\right)\right). \tag{19}$$

Then two user embeddings $\mathbf{u}_i^{r(s)}$ and $\mathbf{u}_{i(J)}^{r(n)}$ are fused by an attention mechanism as follows:

$$\mathbf{u}_i^{r(f)} = \alpha\mathbf{u}_i^{r(s)} + \beta\mathbf{u}_{i(J)}^{r(n)}, \tag{20}$$

where α and β are defined as:

$$\alpha = \frac{exp\left(dot\left(\mathbf{u}_i^{r(s)}, \mathbf{e}_o^r\right)\right)}{exp\left(dot\left(\mathbf{u}_i^{r(s)}, \mathbf{e}_o^r\right)\right) + exp\left(dot\left(\mathbf{u}_{i(J)}^{r(n)}, \mathbf{e}_o^r\right)\right)} = 1 - \beta. \tag{21}$$

Finally, we utilize a multi-layer perceptron (MLP) to output the final prediction $\hat{r}_{io}$ as follows:

$$\hat{r}_{io} = \mathcal{M}\left(\mathcal{M}\left(\cdots\mathcal{M}\left(\left[\mathbf{u}_i^{r(f)},\ \mathbf{e}_o^r\right]\right)\right)\right) = \mathcal{M}_H\left(\left[\mathbf{u}_i^{r(f)},\ \mathbf{e}_o^r\right]\right), \tag{22}$$

where $\mathcal{M}(x) = \mathcal{A}(wx + b)$ is the non-linear transformation for vector x (with weight matrix w and bias vector b), and H is the number of layers in the MLP. If the user-item interaction matrix R is constructed as an implicit feedback matrix, $\hat{r}_{io}$ will be further operated by the sigmoid function to obtain the preference probability from user u_i to item v_o.

2.3.2 Social network embedding task

Such a task is used for link prediction which predicts the trust relationships between users. We encode the social relations into low-dimensional representations by using an

embedding-based method. For a relationship between user u_i and user u_j, we obtain their recommendation enhanced social embedding $\mathbf{u}_i^{s(r)}$ and $\mathbf{u}_j^{s(r)}$ through knowledge assistance modules:

$$\mathbf{u}_i^{s(r)} = \mathcal{K}_L\left(\mathbf{u}_i^r, \mathbf{u}_i^s\right)\left\langle \mathbf{u}_i^s \right\rangle, \tag{23}$$

$$\mathbf{u}_j^{s(r)} = \mathcal{K}_L\left(\mathbf{u}_j^r, \mathbf{u}_j^s\right)\left\langle \mathbf{u}_j^s \right\rangle. \tag{24}$$

Finally, leveraging two MLPs, the probability of the trust relationship $\hat{s}_{ij}$ between user u_i and user u_j is calculeted as follows:

$$\hat{s}_{ij} = S\left(\left[\mathcal{M}_H\left(\mathbf{u}_i^{s(r)}\right), \mathcal{M}'_H\left(\mathbf{u}_j^{s(r)}\right)\right]\right), \tag{25}$$

where S is the similarity function between two real feature vectors. In this paper, we use the cosine function $S(a,b) = \frac{dot(a,b)}{\|a\|\|b\|}$ as our similarity function. One can design other similarity metrics here such as normalized inner product $S(a,b) = \sigma(a^\top b)$.

2.4 Knowledge-aware recommendation

In knowledge-aware recommendation, we design a knowledge graph embedding task to assist the recommendation task.

2.4.1 Recommendation task

Similar to the recommendation task in social-aware recommendation, such task predicts rating or preference probability $\hat{r}_{io}$ from user u_i to item v_o. We first learn two item embeddings: knowledge enhanced embedding $\mathbf{e}_o^{r(k)}$ and neighbor enhanced embedding $\mathbf{e}_{o(J)}^{r(n)}$ as follows:

$$\mathbf{e}_o^{r(k)} = \mathcal{K}_L\left(\mathbf{e}_o^r, \mathbf{e}_o^k\right)\left\langle \mathbf{e}_o^r \right\rangle, \tag{26}$$

$$\mathbf{e}_{o(J)}^{r(n)} = \mathcal{A}\left(w_{(J-1)}\left(\mathbf{e}_{o(J-1)}^r + \sum_{e_p \in nei^s(v_o)} att(v_o, e_p)\mathbf{e}_{p(J-1)}^r\right)\right), \tag{27}$$

where $nei^s(v_o)$ is the v_o's sampled fixed-size neighbor set in the knowledge graph, where $|nei^s(v_o)| = n_k$. att is the function to study the importance of different neighbors for the given item v_o, which is defined as follows:

$$att(v_o, v_p) = \frac{exp\left(dot\left(\mathbf{e}_o^r, \mathbf{e}_p^r\right)\right)}{\sum_{e_q \in nei(v_o)} exp\left(dot\left(\mathbf{e}_o^r, \mathbf{e}_q^r\right)\right)}. \tag{28}$$

Then we utilize an attention mechanism to weigh two item embeddings $\mathbf{e}_o^{r(k)}$ and $\mathbf{e}_{o(J)}^{r(n)}$ as follows:

$$\mathbf{e}_o^{r(f)} = \alpha \mathbf{e}_o^{r(s)} + \beta \mathbf{e}_{o(J)}^{r(n)}, \tag{29}$$

where α and β are defined as:

$$\alpha = \frac{exp\left(dot\left(\mathbf{u}_i^r, \mathbf{e}_o^{r(s)}\right)\right)}{exp\left(dot\left(\mathbf{u}_i^r, \mathbf{e}_o^{r(s)}\right)\right) + exp\left(dot\left(\mathbf{u}_i^r, \mathbf{e}_{o(J)}^{r(n)}\right)\right)} = 1 - \beta. \tag{30}$$

Finally, we utilize a MLP to predict final ratings $\hat{r}_{io}$ from user u_i to item v_o as follows:

$$\hat{r}_{io} = \mathcal{M}_H \left(\left[\mathbf{u}_i^r, \ \mathbf{e}_o^{r(f)} \right] \right). \tag{31}$$

Note that if we focus on users' implicit feedback to items, $\hat{r}_{io}$ will be further operated by the sigmoid function to obtain the preference probability from user u_i to item v_o.

2.4.2 Knowledge graph embedding task

Knowledge graph embedding methods can be categorized into two groups: translational distance models and semantic matching models [57]. We utilize a deep semantic matching method for the triple classification task which predicts whether a triple is correct or not. Note that, one can employ or redesign other knowledge graph embedding methods, such as SLM [49] and SME [4]. For a triple $\left(e_o, r_w, e_p\right)_i$, we first utilize knowledge assistance modules to obtain recommendation enhanced entity embedding $\mathbf{e}_o^{k(r)}$ and $\mathbf{e}_p^{k(r)}$:

$$\mathbf{e}_o^{k(r)} = \mathcal{K}_L \left(\mathbf{e}_o^r, \mathbf{e}_o^k \right) \left\langle \mathbf{e}_o^k \right\rangle, \tag{32}$$

$$\mathbf{e}_p^{k(r)} = \mathcal{K}_L \left(\mathbf{e}_p^r, \mathbf{e}_p^k \right) \left\langle \mathbf{e}_p^k \right\rangle. \tag{33}$$

Then we integrate relation r_w in our method and use MLPs to predicte probability $s\hat{c}o_i$:

$$s\hat{c}o_i = \sigma \left(\mathcal{M}_H'' \left(\mathcal{M}_H \left(\mathbf{e}_o^{k(r)} \right) \odot \mathbf{r}_w \odot \mathcal{M}_H' \left(\mathbf{e}_p^{k(r)} \right) \right) \right). \tag{34}$$

We consider $s\hat{c}o_i$ as the score of the triple $\left(e_o, r_w, e_p\right)_i$ which indicates the plausibility of the fact in the knowledge graph.

2.5 Optimization

To estimate model parameters, we have two following objective functions for GSIRec according to kinds of side information:

$$\mathcal{L}_{OSS}^S = \mathcal{L}_{REC} + \gamma_s \mathcal{L}_{SNE} + \lambda \left\| \Theta \right\|_2^2, \tag{35}$$

$$\mathcal{L}_{OSS}^K = \mathcal{L}_{REC} + \gamma_k \mathcal{L}_{KGE} + \lambda \left\| \Theta \right\|_2^2. \tag{36}$$

$\mathcal{L}_{OSS}^S$ and $\mathcal{L}_{OSS}^K$ are for social-aware recommendation and knowledge-aware recommendation, respectively. $\mathcal{L}_{REC}$ measures the loss in the recommendation task. Depending on the ways of constructing of the user-item interaction matrix $\mathbf{R}$ (rating score or implicit feedback), we use different loss functions for $\mathcal{L}_{REC}$, as follows:

$$\mathcal{L}_{REC}^1 = \sum_{(i,j) \in \mathcal{O}_r} \left(r_{ij} - \hat{r}_{ij} \right)^2, \tag{37}$$

$$\mathcal{L}_{REC}^2 = - \sum_{(u_i, v_j, v_{\tilde{j}}) \in \mathcal{D}_r} \left(r_{ij} \log(\hat{r}_{ij}) + \left(1 - r_{i\tilde{j}} \right) \log \left(1 - \hat{r}_{i\tilde{j}} \right) \right). \tag{38}$$

The first one is for the rating prediction task, where $\mathcal{O}_r$ denotes the observed values in $\mathbf{R}$. The second is for recommendation with implicit feedback. We denotes $\mathcal{I}_i$ is the item set which user i has interacted with, and $\mathcal{D}_r$ can be defined as:

$$\mathcal{D}_r = \left\{ \left(u_i, v_j, v_{\tilde{j}} \right) \mid u_i \in U \wedge v_j \in \mathcal{I}_i \wedge v_{\tilde{j}} \in V \backslash \mathcal{I}_i \right\}. \tag{39}$$

$\mathcal{L}_{SNE}$ and $\mathcal{L}_{KGE}$ measure the loss in the social network embedding task and knowledge graph embedding task, respectively:

$$\mathcal{L}_{SNE} = -\sum_{(u_a,u_b,u_{\tilde{b}})\in\mathcal{D}_s} \left(s_{ab}\log\left(\hat{s}_{ab}\right) + \left(1-s_{a\tilde{b}}\right)\log\left(1-\hat{s}_{a\tilde{b}}\right)\right), \tag{40}$$

$$\mathcal{L}_{KGE} = -\left(\sum_{(h,r,t)_i\in\mathbf{G}} \hat{sco}_i - \sum_{(h,r,t)_j\notin\mathbf{G}} \hat{sco}_j\right). \tag{41}$$

For $\mathcal{L}_{SNE}$, it uses binary cross-entropy function to calculate the loss, where $\mathcal{D}_s$ is defined as:

$$\mathcal{D}_s = \left\{(u_a, u_b, u_{\tilde{b}}) \mid u_a \in U \wedge u_b \in nei(u_a) \wedge u_{\tilde{b}} \in U\backslash nei(u_a)\right\}. \tag{42}$$

As for $\mathcal{L}_{KGE}$, we let observed triples in knowledge graph $\mathbf{G}$ tend to have higher scores and decreasing the scores for all unobserved triples.

The last term in loss function is the L2 regularization term to control the model complexity and avoid over-fitting. γ_s, γ_k and λ are the balancing parameters in the objective function.[1] We utilize an alternating strategy to optimize our framework, which enables the whole framework to be more flexible. The overall logic of GSIRec is summarized in Algorithm 1.

Algorithm 1 Training algorithm of GSIRec.

Input: Interaction matrix $\mathbf{R}$, side information (social network $\mathbf{S}$ or knowledge graph $\mathbf{G}$)
Output: Prediction function $\mathcal{F}(u, v|\Theta, \mathbf{R}, \mathbf{S})$ or $\mathcal{F}(u, v|\Theta, \mathbf{R}, \mathbf{G})$

```
Initialize all parameters.
for number of training iterations do
    sample batch b_s of positive and negative data from S or G
    calculate the L_SNE or L_KGE and update parameters of F
    sample batch b_r of interaction data from R
    calculate the L_REC and update parameters of F
end for
```

3 Experiment

In this section, we conduct extensive experiments to evaluate our GSIRec framework through addressing the following research questions:

- (**Q**1) Compared with the state-of-the-art baselines, how does our framework GSIRec perform?
- (**Q**2) Can GSIRec effectively alleviate data sparsity and cold start issues?
- (**Q**3) Do the settings of key hyper-parameters and designs affect the performance of GSIRec?
- (**Q**4) Can the recommendation task be helpful for the side information task in GSIRec?

[1] Following the related work [64], γ. can be seen as the ratio of two learning rates for the recommendation task and side information task.

In what follows, we first introduce the experimental settings, then answer the above four questions.

3.1 Experiment setup

In this subsection, we introduce the datasets, baselines, the choice of hyper-parameters, and evaluation protocols.

3.1.1 Datasets

Four datasets Ciao,[2] Epinions,[3] DBbook,[4] MovieLens[5] are used in our experiments. Ciao and Epinions are for social-aware recommendation, and DBbook and MovieLens are used for knowledge-aware recommendation.

- Ciao dataset was obtained from a real-world social media website. The dataset contains both users' explicit ratings (ranging from 1 to 5) on items and users' social relations.
- Epinions dataset was obtained from a popular review website. The dataset contains rating data (ranging from 1 to 5) and user-user trust relationships.
- DBbook is a public dataset in book recommendations, which contains users' explicit ratings (ranging from 1 to 5) on books. The knowledge graph for DBbook is released by [5].
- MovieLens dataset is a widely used movie dataset and it contains users' explicit feedbacks (ranging from 1 to 5). The knowledge graph MovieLens is released by [64].

In the experiment, we evaluate our method GSIRec in both rating prediction task and top-K recommendation task. Similar to the related work [61, 64], for top-K recommendation, we transform the explicit rating as implicit feedback indicating that the user has rated the item, and generate a negative sample set for each user, which is of equal size with the rated ones. The statistics of the four datasets are summarized in Table 2.

3.1.2 Baselines

We apply GSIRec in social-aware recommendation and knowledge-aware recommendation. Therefore, we compare GSIRec with the following four classes of baselines: (i) pure collaborative filtering method GCMC [2]; (ii) feature enhanced methods including NFM [20] and DeepFM [16]; (iii) social enhanced methods which take users' social influences into consideration including DeepSoR [14], GraphRec [15] and DiffNet [67]; (iv) knowledge graph enhanced methods which take knowledge graph into consideration including KGCN [65], MKR [64] and KGAT [55]. The characteristics of the comparison methods are listed as follows:

- GCMC is a graph-based recommendation framework, which adopts a graph auto-encoder in the user-item bipartite graph to learn user and item embeddings for rating prediction. In our experiment, we treat GCMC as a plain collaborative filtering method that only leverages user-item interactions for recommendation.

[2]Ciao: http://www.cse.msu.edu/~tangjili/index.html
[3]Epinions: http://www.cse.msu.edu/~tangjili/index.html
[4]DBbook: http://2014.eswc-conferences.org/important-dates/call-RecSys.html
[5]MovieLens: https://grouplens.org/datasets/movielens/1m/

Table 2 Basic statistics of four datasets. The "graph links" in Ciao and Epinions means social links of users in social networks, and the "graph links" in DBbook and MovieLens means item links in knowledge graphs

Dataset	# Users	# Items	# Interactions	# Graph links
Ciao	7,317	104,975	281,898	85,205
Epinions	18,097	261,679	756,678	287,081
DBbook	5,576	2,680	65,961	133,529
MovieLens	6,040	2,347	566,461	20,195

- NFM is a factorization model that improves FM [44] by using a neural network to capture high-order feature interaction. Here we utilize all the side information as additional input features. Specifically, we concatenate the user embedding, item embedding, and the average embeddings of user neighbors in the social network (or average embeddings of item neighbors in the knowledge graph) as the inputs.
- DeepFM is a general feature enhanced factorization model, which combines factorization machines and deep neural networks for recommendation. We provide the same inputs as NFM for DeepFM.[6]
- DeepSoR combines a neural network to learn latent preferences of users from social networks with PMF [42].
- GraphRec is a state-of-the-art social recommender, which considers the user-item graph and user-user graph for social recommendation.
- DiffNet is a state-of-the-art method for social recommendation, which utilizes graph convolutional networks and designs layer-wise influence diffusion structure for users to learn user embeddings.
- KGCN is a state-of-the-art model for knowledge-aware recommendation, which uses graph attention convolutional methods to capture semantic information in the knowledge graph.
- MKR is a state-of-the-art multi-task learning framework, which employs knowledge graph embedding tasks to enhance collaborative filtering tasks.
- KGAT is a state-of-the-art propagation-based model, which utilizes graph attention networks for knowledge-aware recommendation. Compared with KGCN, KGAT combines the user-item interaction graph and knowledge graph as a unified graph for feature learning.
- GSIRec(s)-a is our model with attention-based knowledge assistance modules for social-aware recommendation.
- GSIRec(s)-g is our model with gate-based knowledge assistance modules for social-aware recommendation.
- GSIRec(k)-a is our model with attention-based knowledge assistance modules for knowledge-aware recommendation.
- GSIRec(k)-g is our model with gate-based knowledge assistance modules for knowledge-aware recommendation.

[6]We have tried other factorization models FM [44] and Wide&Deep [13], and find that NFM and DeepFM are slightly better than them. Therefore, we present the better one here.

3.1.3 Parameter settings

We implemented GSIRec framework with Pytorch which is a Python library for deep learning. For each dataset, we randomly split it into training, validation, and test sets following 6 : 2 : 2. We repeated each experiment 5 times and reported the average performance. The model parameters are first initialized according to the Gaussian distribution. We utilized ReLU as the activation function and Adam [28] for optimization with an initial learning rate of 0.001. For the numbers of layer J in neighbor aggregation operation, we find that when J=1 is good enough and similar results can be found in many other studies [2, 65]. The hyper-parameters were tuned on the validation set. The coefficient of L2 regularization λ was tuned in [10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}]. For all neural components, we implement two hidden layers by default. In addition, we sampled one negative instance per positive instance in the social network embedding task or knowledge graph embedding task. Table 3 shows our hyper-parameters settings. The parameters of the baselines are carefully tuned to achieve the best performance for fair comparisons. We will study the impact of hyper-parameters in the following subsection.

3.1.4 Evaluation protocols

We evaluate our method GSIRec in the rating prediction task and top-K recommendation. For the rating prediction task, we adopt *Root Mean Square Error* (RMSE) and *Mean Absolute Error* (MAE) to evaluate the performance of the methods. Smaller values of these two metrics indicate the better recommendation. For the top-K recommendation, we choose Precision@K and Recall@K to evaluate. We perform item ranking on all items to guarantee that the evaluation process is unbiased [30]. In our experiment, we set K = 5.

3.2 Performance comparison (Q1)

Tables 4 and 5 show the performance of all compared methods in social-aware recommendation. Tables 6 and 7 show the performance of all compared methods in knowledge-aware recommendation respectively. ** denotes the best values among all methods, and * denotes the best values among all baselines. According to the results, we have the following main observations:

- Feature enhanced methods NFM and DeepFM outperform pure collaborative filtering method GCMC in most cases, which indicates that side information is complementary to user-item interaction data in recommendations.
- Since DeepSoR, GraphRec, and DiffNet utilize social information to improve recommendation, they outperform pure collaborative filtering model GCMC. In addition, GraphRec and DiffNet generally achieve better performance than other baselines

Table 3 Hyper-parameter settings for four datasets

Dataset	Hyper-parameter
Ciao	$L = 2, d = 16, b_r = 1024, b_s = 1024, n_s = 7, \lambda = 10^{-5}, \gamma = 0.2$
Epinions	$L = 3, d = 16, b_r = 1024, b_s = 1024, n_s = 7\ \lambda = 10^{-5}, \gamma = 0.1$
DBook	$L = 1, d = 8, b_r = 512, b_s = 2048, n_k = 5, \lambda = 10^{-4}, , \gamma = 0.5$
MovieLens	$L = 1, d = 8, b_r = 2048, b_s = 2048, n_k = 5, \lambda = 10^{-6}, \gamma = 0.1$

Table 4 The results of MAE and RMSE in social-aware recommendation

Method	Ciao		Epinions	
	MAE	RMSE	MAE	RMSE
GCMC	0.8113	1.0350	0.8649	1.1046
NFM	0.8083	1.0582	0.8531	1.0968
DeepFM	0.8107	1.0461	0.8499	1.0985
DeepSoR	0.7787	1.0395	0.8537	1.1001
GraphRec	0.7683*	1.0185*	0.8426	1.0854*
DiffNet	0.7706	1.0206	0.8375*	1.0896
GSIRec(s)-a	0.7627	1.0018	0.8233**	1.0808
GSIRec(s)-g	0.7601**	1.0004**	0.8250	1.0789**

in social-aware recommendation in most cases. These results imply that the graph convolutional methods are powerful for graph side information recommendation.

– For knowledge-aware recommendation, KGCN and KGAT stronger than other baselines, indicating the power of graph convolutional networks in graph data. In addition, MKR performs much better than GCMC, which suggests the cross-compress units capture useful knowledge in knowledge graph embedding tasks.
– GSIRec leverages multi-task learning and graph attention method to learn user and item representations for recommendation. In general, GSIRec with gate-based knowledge assistance modules stronger than GSIRec with attention-based knowledge assistance modules. Without special mention, we show the results of GSIRec with gate-based knowledge assistance modules in the following experiment. In both rating prediction task and top-K recommendation, our method GSIRec achieves the best results on all metrics compared with other methods. Specifically, for the rating prediction task, GSIRec improves over the state-of-the-art baselines w.r.t. RMSE by 1.81%, 0.60%, 0.97% and 1.05% in Ciao, Epinions, DBbook, and MovieLens respectively. For the top-K recommendation, GSIRec improves over the strongest baselines w.r.t. Precision by 3.17%, 4.06%, 0.99% and 4.12% in Ciao, Epinions, DBbook, and MovieLens respectively.

Table 5 The results of Precision and Recall in social-aware recommendation

Method	Ciao (@5, %)		Epinions (@5, %)	
	Precision	Recall	Precision	Recall
GCMC	1.7576	1.3090	0.8427	0.7885
NFM	1.8108	1.3223	0.8857	0.7705
DeepFM	1.8118	1.3424	0.8578	0.7875
DeepSoR	1.8015	1.3097	0.9111*	0.8128
GraphRec	1.8795	1.3371	0.8895	0.7897
DiffNet	1.8911*	1.3847*	0.9017	0.8201*
GSIRec(s)-a	1.9323	1.4331	0.9359	0.8487
GSIRec(s)-g	1.9531**	1.4790**	0.9483**	0.8537**

Table 6 The results of MAE and RMSE in knowledge-aware recommendation

Method	DBbook		MovieLens	
	MAE	RMSE	MAE	RMSE
GCMC	0.7907	0.9963	0.7369	0.9260
NFM	0.7892	0.9916	0.7273	0.9215
DeepFM	0.7867	0.9874	0.7204	0.9159
KGCN	0.7465*	0.9543	0.7193	0.9114*
MKR	0.7546	0.9572	0.7277	0.9196
KGAT	0.7528	0.9384*	0.7162*	0.9189
GSIRec(k)-a	0.7362**	0.9294**	0.7117	0.9035
GSIRec(k)-g	0.7371	0.9375	0.7093**	0.9019**

3.3 Cold start and data sparsity issues (Q2)

As mentioned in the introduction section, data sparsity and cold start are two challenges faced by most recommenders. In this subsection, we investigated the ability of our model in handling these two issues.

3.3.1 Results in data sparse scenarios

We first evaluate the capabilities of addressing data sparsity by respective competitors. The experiments were conducted by utilizing different ratios (from 80% to 20%) of the training set on four datasets and keeping fixed the validation and test set during the experiment. Figures 4 and 5 show the results of all methods with respect to different proportions of the training set in social-aware recommendation and knowledge-aware recommendation, respectively. For all datasets, we observe that as the increase of sparsity level, the overall performance decreases among all models. Our method GSIRec outperforms other baselines in most cases. It is worth mentioning that our method consistently outperforms all baselines when the ratio is 20%, which verifies that GSIRec can maintain a good performance when data are extremely sparse.

Table 7 The results of Precision and Recall in knowledge-aware recommendation

Method	DBbook (@5, %)		MovieLens (@5, %)	
	Precision	Recall	Precision	Recall
GCMC	1.7592	3.5956	15.723	4.7338
NFM	2.0102	3.7497*	19.473*	6.0550*
DeepFM	1.9367	3.7129	19.400	5.9934
KGCN	1.9702	3.7364	17.766	5.6171
MKR	1.8927	3.5614	17.979	5.6484
KGAT	2.0314*	3.7072	18.968	5.8224
GSIRec(k)-a	2.0441	3.8011	20.242**	6.2592**
GSIRec(k)-g	2.0565**	3.9968**	19.936	6.0888

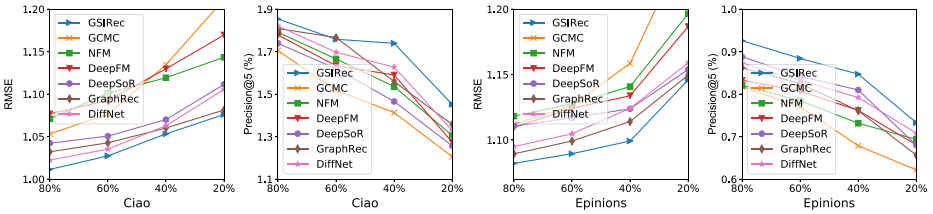


Figure 4 The results for social-aware recommendation on Ciao and Epinions with different ratios (from 80% to 20%) of training set

3.3.2 Results in cold start scenarios

According to the side information of the datasets, we consider addressing the cold-start user problem on Ciao and Epinions datasets, and addressing the cold-start item problem on DBbook and MovieLens datasets. We treat those who have rated x or fewer ratings as cold start users and those that have been rated less than x as cold start items. Followed by other related works [25, 35], we set x=5. Figure 6 illustrates the RMSE results of our method GSIRec and several strong baselines in cold-start scenarios on four datasets. We can see that our method GSIRec is beneficial to the relatively inactive users and items in recommendation scenarios.

3.4 Model analysis (Q3)

In order to gain more insight into our method, we study an in-depth model analysis in this subsection. We first investigate the effect of the knowledge assistance module, then analyze the sensitivity of key hyper-parameters, and finally explore whether the combination of the user side information task and item side information task can enhance the performance.

3.4.1 Effect of knowledge assistance module

In order to further explore the utility of the knowledge assistance module, we replace our knowledge assistance module with other four feature enhanced modules: cross-stitch unit [41], cross-connection unit [24], cross-compress unit [64], and feature evolution unit [66]. These feature enhanced units are utilized to be the connection between the recommendation task and the side information task for multi-task feature learning, and they are related to our knowledge assistance modules. We will further discuss them in the related work section. The comparison results are shown in Figures 7 and 8. According to the results, our

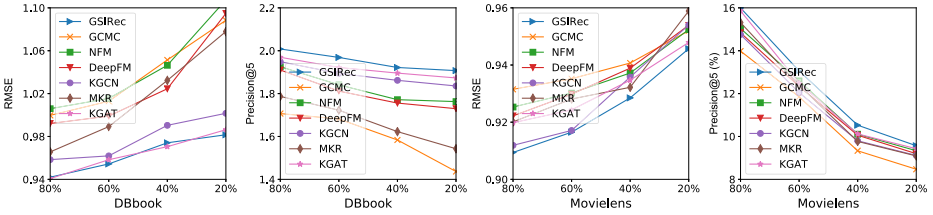


Figure 5 The results for knowledge-aware recommendation on DBbook and MovieLens with different ratios (from 80% to 20%) of training set

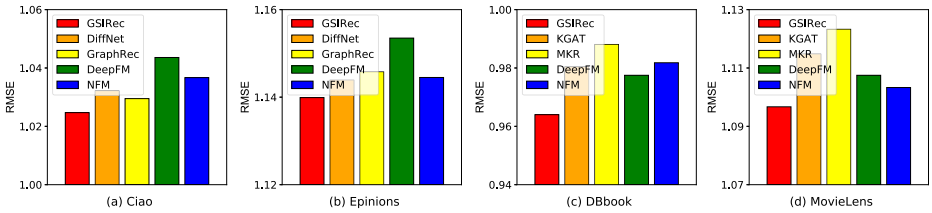


Figure 6 The results of RMSE on four datasets in cold start scenarios

proposed feature enhanced module achieves the best results indicates that our knowledge assistance modules can capture useful knowledge from the side information task to assist the recommendation.

3.4.2 Effect of hyper-parameters

We explore the effect of two hyper-parameters: embedding size d and neighbor sample size n_s(or n_k) in side information tasks. The results are shown in Figures 9 and 10. We have the following observations: (i) In general, we find that the performance is improved as the embedding size d increases because large embedding sizes mean more expressive. However, when d further increases, our methods degrade the performance, because a too large dimension increases the complexity of our method and may overfit the datasets. (ii) In addition, we find that our method achieves the best performance when $n_s = 7$ in social-aware recommendation and $n_k = 5$ knowledge-aware recommendation. The reason is that a too small neighbor sample size does not have enough capacity to aggregate neighbor information, while a too large neighbor sample size may introduce noises.

3.4.3 Results on the unified framework

GSIRec is designed for side information recommendation. In the above experiments, we have validated the effectiveness of GSIRec in social-aware recommendation and knowledge-aware recommendation. In this subsection, we would like to apply our method to the scenario where both users and items are associated with the side information. We further explore whether the combination of the user side module and item side module can enhance the performance. For this purpose, we plug the user side module and item side module into the unified framework GSIRec. The objective function for the unified framework GSIRec

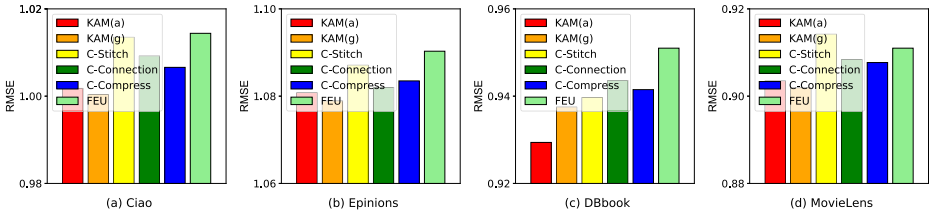


Figure 7 The results of RMSE on GSIRec with different feature enhanced modules. KAM(a): attention-based knowledge assistance module; KAM(g): gate-based knowledge assistance module; C-Stitch: cross-stitch unit; C-Connection: cross-connection unit; C-Compress: cross-compress unit; FEU: feature evolution unit

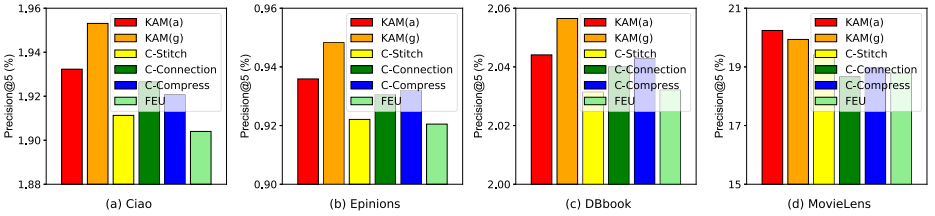


Figure 8 The results of Precision on GSIRec with different feature enhanced modules. KAM(a): attention-based knowledge assistance module; KAM(g): gate-based knowledge assistance module; C-Stitch: cross-stitch unit; C-Connection: cross-connection unit; C-Compress: cross-compress unit; FEU: feature evolution unit

can be defined as follows:

$$\mathcal{L}_{OSS}^{U} = \mathcal{L}_{REC} + \gamma_s \mathcal{L}_{SNE} + \gamma_k \mathcal{L}_{KGE} + \lambda \left\| \Theta \right\|_2^2, \tag{43}$$

where γ_s and γ_k are the balancing parameters for user side information task and item side information task, respectively. Because we didn't find the suitable datasets containing both social networks and knowledge graphs information that can be used for both the rating prediction task and top-K recommendation, we construct top-n semantic friends (implicit social information) for each user in the knowledge-aware datasets (DBbook and MovieLens datasets) according to the approach in [75]. The research reveals that semantic friends (implicit social information) can identify with whom a user has similar preferences [37, 75]. In this way, both users and items in these two datasets are associated with side information. In our experiment, we set $n = 5$, $\gamma_s = 10^{-5}$, $\gamma_k = 10^{-5}$. One can utilize other methods to generate implicit user friends, such as the methods mentioned in [37] and [27]. To evaluate the performance of the unified framework GSIRec, we compare it with its two variants: GSIRec(s) (only using user side information) and GSIRec(k) (only using item side information). Figure 11 shows the performance of different methods. From the results, we find that GSIRec which leverages both user and item side information performs better than two variants GSIRec(s) and GSIRec(k), indicating that leveraging both user side and item side information can enhance the recommendation performance.

3.5 Results on side information tasks (Q4)

The goal of multi-task learning is to utilize useful information contained in related tasks to help enhance the performance of all the tasks [6]. Therefore, we would like to investigate whether the recommendation task benefits the side information tasks in GSIRec. Specifically, we conducted the following experiments of link prediction tasks to show performance

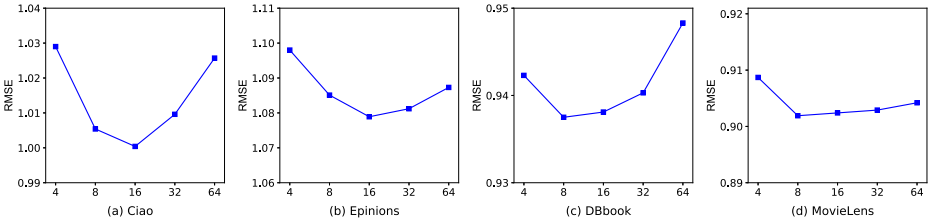


Figure 9 The results of RMSE on GSIRec with different embedding side d

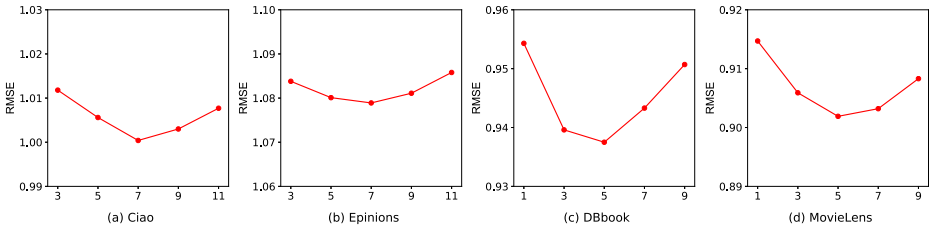


Figure 10 The results of RMSE on GSIRec with different neighbor sample size n_s or n_k

differences between the two settings: (i) only side information task is trained and (ii) side information task and recommendation task are trained together. We split the graph data into training and test sets following 6:4 and use F1-score as the evaluation metric. Figure 12 presents the performance comparison on four datasets. From the results, we find that making use of the recommendation task improves the F1-score of link prediction in the social network embedding task and knowledge graph embedding task, respectively. The results indicate knowledge assistance module can indeed capture useful information from related tasks to assist each task in enhancing its performance.

4 Related work

4.1 Social-aware recommendation

With the prevalence of online social media, many E-commerce sites have become popular social platforms in which users can not only select items they love but also follow other users. According to the social influence theory [1, 3, 31], users' preferences are influenced by their social neighbors. Therefore, researchers propose using social networks as another information stream to mitigate the lack of user-item interaction data and improve recommendation, also known as social recommendation.

The most common approach in social recommendation is to design loss terms for social influence and integrate both social loss and recommendation loss into a unified loss function for jointly optimizing [32, 39, 52]. For instance, SoReg assumes that social neighbors share similar feature representations and design regularization terms in the matrix factorization framework [39]. CSR models the characteristics of social relations and designs a characterized regularization term to improve SoReg [32]. SoDimRec exploits the heterogeneity of social relations and weak dependency connections to regularization terms [52]. In addition to the above-mentioned regularization-based social recommenders, a more explicit

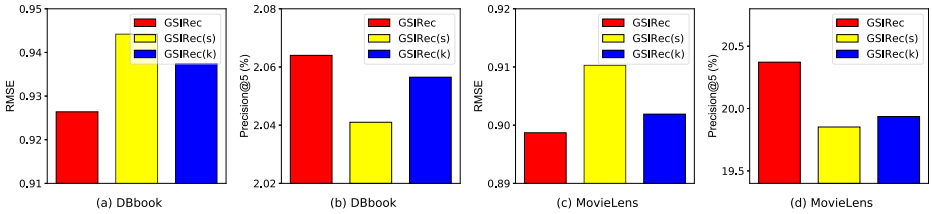


Figure 11 The results of RMSE and Precision on different GSIRec variants

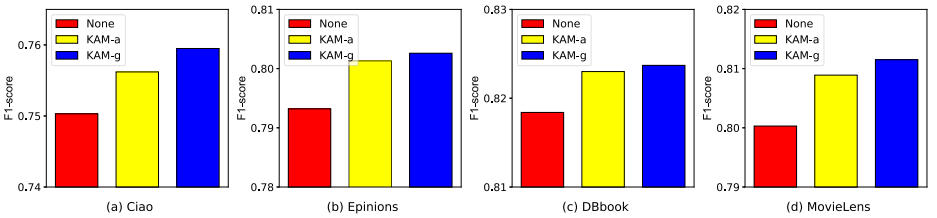


Figure 12 The results of F1-score on the side information task. "None" means only side information task is trained, while "KAM-a" and "KAM-g" means side information task and recommendation task are trained together using attention-based knowledge assistance module and gate-based knowledge assistance module, respectively

and straightforward way is to explicitly models social relations in the predictive model [7, 17, 38]. For example, TrustSVD extended SVD++ [29] by treating the preferences of social neighbors as auxiliary implicit feedbacks [17].

With the development of deep learning, more and more social recommenders utilize neural components to enhance performance. For instance, SAMN considers both aspect-level and friend-level differences and utilizes memory network and attention mechanisms for social recommendation [10]. EATNN utilizes attention mechanisms to assigns a personalized transfer scheme for each user [12]. In recent years, Graph Neural Networks (GNNs) have achieved state-of-the-art performance of graph representation learning. Since the social network can be represented as a user-user social graph, researchers develop social recommenders based on GNNs to combine the user representation with the user's social neighbors [9, 15, 34, 67]. For example, HOSR [34] and DiffNet [67] stack more graph convolutional layers and propagate user embeddings along the social network to capture high-order neighbor information.

In this work, GSIRec utilizes the graph attention method to learn social neighbor enhanced representations in social-aware recommendation. In addition, GSIRec derives a multi-task learning approach that introduces a social network embedding task to assist the recommendation task. The social network embedding task can be seen as the constraint term explicitly to provide regularization for recommender systems.

4.2 Knowledge-aware recommendation

Knowledge-aware recommendation has been shown effective in alleviating data sparsity issues for recommender systems by using the knowledge graph as the side information. The utilization of the knowledge graph can be roughly categorized into three groups [18, 56]: (i) the embedding-based method; (ii) the path-based method; and (iii) the propagation-based method.

The embedding-based method utilizes knowledge graph embedding algorithms to preprocess the knowledge graph and incorporates pre-trained entity embeddings in the recommendation task [5, 62, 64, 76]. For instance, CKE [76] adopts TransR [33] to learn entity embeddings with the involvement of the knowledge graph. KTUP [5] adopts TransH [60] to design a translation-based recommender model that jointly learns the task of recommendation and knowledge graph completion. DKN [62] treats entity embeddings (learned from TransD [26]) and word embeddings as different channels and designs a CNN framework to incorporate knowledge-level and semantic-level representations of news for news recommendation.

The path-based method leverages multiple patterns of connections among entities in knowledge graphs for recommendation [8, 23, 48, 50, 58, 71]. For example, PER [72] and FMG [77] treat knowledge graphs as heterogeneous information networks and design different meta-paths or meta-graphs to extract semantic information for recommendation.

The propagation-based method, which is based on the idea of embedding propagation, considers both semantic information and connectivity information in knowledge graphs [47, 53, 55, 56, 59, 61, 63, 65]. For instance, RippleNet [61] introduces the concept of preference propagation which propagates users' potential preferences along links in knowledge graphs to enrich user representations. KGCN [65] adopts graph convolutional networks to aggregate item neighbor information in the knowledge graph to enhance item representations. KGCN-LS [63] further enhances the performance of KGCN [65] by adding a label smoothness mechanism. KGAT [55] combines the user-item interaction graph and knowledge graph as a unified graph and utilizes graph attention networks for feature learning. KGPolicy [59] adopts graph convolutional networks to prepare high-quality representations of nodes in knowledge graphs and uses reinforcement learning to explore high-quality negative items for positive user-item interactions. CKAN [56] explicitly encodes the collaborative signals by embedding propagation and combine the collaborative signals with knowledge associations in an end-to-end manner.

Our method GSIRec is a hybrid method that combines the embedding-based method and propagation-based method. GSIRec designs the knowledge assistance module to capture useful knowledge from the knowledge graph embedding task. In addition, GSIRec utilizes a graph attention method to exploit semantic information from knowledge graphs.

4.3 Multi-task learning for recommendation

Multi-task learning (MTL) is an approach that improves learning for one task by using the knowledge contained in the training signals of other related tasks [6]. Recently, many recommenders leverage MTL to improve recommendation performance [5, 12, 24, 64, 66]. KTUP [5] jointly trains the tasks of recommendation and knowledge graph completion and both tasks show excellent performance. JNSKR [11] designs an efficient negative sampling optimization for knowledge-aware recommendation which integrates the recommendation task and the knowledge graph embedding task in a multi-task learning framework. MKR [64] designs a cross&compress unit to be the connection between the collaborative filtering task and knowledge graph embedding task, and two tasks share their latent features in the cross&compress unit for knowledge-aware recommendation. CoNet [24] replaces the trainable transfer scalars in cross-stitch units [41] with trainable transfer matrices, and designs the cross-connection unit to transfer the knowledge between the source task and target task. EATNN [12] integrates both the subtasks of item domain and social domain into a unified multi-task learning framework for social-aware recommendation. TrustEV [66] designs feature evolution units to enable the user social embeddings and user recommendation embeddings to exchange their features for social recommendation.

In this work, we proposed a multi-task learning framework GSIRec that can be applied in social-aware recommendation and knowledge-aware recommendation. We proposed completely different knowledge assistance modules from the above-mentioned transfer units. The knowledge assistance modules use attention mechanisms and gated structures to capture useful knowledge from related tasks for enhancing the performance of each task.

5 Conclusions and future work

In this work, we advance the graph side information recommendation. We presented a novel framework called GSIRec, which can leverage multi-task learning and graph neural networks to enhance the recommendation task. Extensive experimental results on four real-world datasets demonstrate GSIRec not only outperforms the state-of-the-art recommendation solutions but also has advantages in handling data sparsity and cold start issues. For future work, we plan to (i) investigate GSIRec's theoretical performance thoroughly; and (ii) do the running time analysis of GSIRec.

Acknowledgments This work was supported by the National Natural Science Foundation of China under Grant No. 61876069; Jilin Province Key Scientific and Technological Research and Development Project under Grant Nos. 20180201067GX and 20180201044GX; and Jilin Province Natural Science Foundation under Grant No. 20200201036JC.

Declarations

Competing interests The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Anagnostopoulos, A., Kumar, R., Mahdian, M.: Influence and correlation in social networks. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 7–15 (2008)
2. Berg, R.V.D., Kipf, T.N., Welling, M.: Graph convolutional matrix completion. arXiv:1706.02263 (2017)
3. Bond, R.M., Fariss, C.J., Jones, J.J., Kramer, A.D., Marlow, C., Settle, J.E., Fowler, J.H.: A 61-million-person experiment in social influence and political mobilization. Nature **489**(7415), 295–298 (2012)
4. Bordes, A., Glorot, X., Weston, J., Bengio, Y.: A semantic matching energy function for learning with multi-relational data. Mach. Learn. **94**(2), 233–259 (2014)
5. Cao, Y., Wang, X., He, X., Hu, Z., Chua, T.S.: Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In: The World Wide Web Conference, pp. 151–161 (2019)
6. Caruana, R.: Multitask learning. Mach. Learn. **28**(1), 41–75 (1997)
7. Chaney, A.J., Blei, D.M., Eliassi-Rad, T.: A probabilistic model for using social networks in personalized item recommendation. In: Proceedings of the 9th ACM Conference on Recommender Systems, pp. 43–50 (2015)
8. Chen, H., Li, Y., Sun, X., Xu, G., Yin, H.: Temporal meta-path guided explainable recommendation. In: WSDM, pp. 1056–1064 (2021)
9. Chen, H., Yin, H., Chen, T., Wang, W., Li, X., Hu, X.: Social boosted recommendation with folded bipartite network embedding. IEEE Trans. Know. Data Eng. (2020)
10. Chen, C., Zhang, M., Liu, Y., Ma, S.: Social attentional memory network: Modeling aspect-and friend-level differences in recommendation. In: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, pp. 177–185 (2019)
11. Chen, C., Zhang, M., Ma, W., Liu, Y., Ma, S.: Jointly non-sampling learning for knowledge graph enhanced recommendation. In: SIGIR (2020)
12. Chen, C., Zhang, M., Wang, C., Ma, W., Li, M., Liu, Y., Ma, S.: An efficient adaptive transfer neural network for social-aware recommendation. In: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 225–234 (2019)
13. Cheng, H.T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., et al: Wide & deep learning for recommender systems. In: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, pp. 7–10 (2016)
14. Fan, W., Li, Q., Cheng, M.: Deep modeling of social relations for recommendation. In: Thirty-Second AAAI Conference on Artificial Intelligence, pp. 8075–8076 (2018)

15. Fan, W., Ma, Y., Li, Q., He, Y., Zhao, E., Tang, J., Yin, D.: Graph neural networks for social recommendation. In: The World Wide Web Conference, pp. 417–426 (2019)
16. Guo, H., Tang, R., Ye, Y., Li, Z., He, X.: DeepFM: a factorization-machine based neural network for CTR prediction. arXiv:1703.04247 (2017)
17. Guo, G., Zhang, J., Yorke-Smith, N.: Trustsvd: Collaborative filtering with both the explicit and implicit influence of user trust and of item ratings. In: AAAI, pp. 123–125 (2015)
18. Guo, Q., Zhuang, F., Qin, C., Zhu, H., Xie, X., Xiong, H., He, Q.: A survey on knowledge graph-based recommender systems. arXiv:2003.00911 (2020)
19. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems, pp. 1024–1034 (2017)
20. He, X., Chua, T.S.: Neural factorization machines for sparse predictive analytics. In: Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval, pp. 355–364 (2017)
21. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.S.: Neural collaborative filtering. In: Proceedings of the 26th International Conference on World Wide Web, pp. 173–182 (2017)
22. He, X., Zhang, H., Kan, M.Y., Chua, T.S.: Fast matrix factorization for online recommendation with implicit feedback. In: Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, pp. 549–558 (2016)
23. Hu, B., Shi, C., Zhao, W.X., Yu, P.S.: Leveraging meta-path based context for top-n recommendation with a neural co-attention model. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1531–1540 (2018)
24. Hu, G., Zhang, Y., Yang, Q.: Conet: Collaborative cross networks for cross-domain recommendation. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, pp. 667–676 (2018)
25. Jamali, M., Ester, M.: A matrix factorization technique with trust propagation for recommendation in social networks. In: Proceedings of the Fourth ACM Conference on Recommender Systems, pp. 135–142 (2010)
26. Ji, G., He, S., Xu, L., Liu, K., Zhao, J.: Knowledge graph embedding via dynamic mapping matrix. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7Th International Joint Conference on Natural Language Processing, pp. 687–696 (2015)
27. Jiang, X., Hu, B., Fang, Y., Shi, C.: Multiplex Memory Network For Collaborative Filtering. In: Proceedings of the 2020 SIAM International Conference on Data Mining, pp. 91–99 (2020)
28. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv:1412.6980 (2014)
29. Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 426–434 (2008)
30. Krichene, W., Rendle, S.: On sampled metrics for item recommendation. In: KDD, pp. 1748–1757 (2020)
31. Lewis, K., Gonzalez, M., Kaufman, J.: Social selection and peer influence in an online social network. Proc. Natl. Acad. Sci. **109**(1), 68–72 (2012)
32. Lin, T.H., Gao, C., Li, Y.: Recommender systems with characterized social regularization. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, pp. 1767–1770 (2018)
33. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. Google Scholar Google Scholar Digital Library Digital Library (2015)
34. Liu, Y., Chen, L., He, X., Peng, J., Zheng, Z., Tang, J.: Modelling high-order social relations for item recommendation. arXiv:2003.10149 (2020)
35. Liu, B.Y.Y.L.D., Liu, J.: Social collaborative filtering by trust. In: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence. IEEE (2013)
36. Long, M., Cao, Z., Wang, J., Philip, S.Y.: Learning multiple tasks with multilinear relationship networks. In: Advances in Neural Information Processing Systems, pp. 1594–1603 (2017)
37. Ma, H.: An experimental study on implicit social recommendation. In: Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 73–82 (2013)
38. Ma, H., King, I., Lyu, M.R.: Learning to recommend with social trust ensemble. In: Proceedings of the 32nd International ACM SIGIR Conference On Research and Development in Information Retrieval, pp. 203–210 (2009)
39. Ma, H., Zhou, D., Liu, C., Lyu, M.R., King, I.: Recommender systems with social regularization. In: Proceedings of the fourth ACM International Conference on Web Search and Data Mining, pp. 287–296 (2011)

40. Massa, P., Avesani, P.: Trust-aware recommender systems. In: Proceedings of the 2007 ACM Conference on Recommender Systems, pp. 17–24 (2007)
41. Misra, I., Shrivastava, A., Gupta, A., Hebert, M.: Cross-stitch networks for multi-task learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3994–4003 (2016)
42. Mnih, A., Salakhutdinov, R.R.: Probabilistic matrix factorization. In: Advances in Neural Information Processing Systems, pp. 1257–1264 (2008)
43. Pan, Y., He, F., Yu, H.: Learning social representations with deep autoencoder for recommender system. World Wide Web pp. 1–21 (2020)
44. Rendle, S.: Factorization machines. In: 2010 IEEE International Conference on Data Mining, pp. 995–1000 (2010)
45. Ruder, S.: An overview of multi-task learning in deep neural networks. arXiv:1706.05098 (2017)
46. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th International Conference on World Wide Web, pp. 285–295 (2001)
47. Sha, X., Sun, Z., Zhang, J.: Attentive knowledge graph embedding for personalized recommendation. arXiv:1910.08288 (2019)
48. Shi, C., Zhang, Z., Ji, Y., Wang, W., Philip, S.Y., Shi, Z.: SemRec: a personalized semantic recommendation method based on weighted heterogeneous information networks. World Wide Web **22**(1), 153–184 (2019)
49. Socher, R., Chen, D., Manning, C.D., Ng, A.: Reasoning with neural tensor networks for knowledge base completion. In: Advances in Neural Information Processing Systems, pp. 926–934 (2013)
50. Sun, Z., Yang, J., Zhang, J., Bozzon, A., Huang, L.K., Xu, C.: Recurrent knowledge graph embedding for effective recommendation. In: Proceedings of the 12th ACM Conference on Recommender Systems, pp. 297–305 (2018)
51. Tang, Y., Guo, K., Zhang, R., Xu, T., Ma, J., Chi, T.: ICFR: An effective incremental collaborative filtering based recommendation architecture for personalized websites. World Wide Web **23**(2), 1319–1340 (2020)
52. Tang, J., Wang, S., Hu, X., Yin, D., Bi, Y., Chang, Y., Liu, H.: Recommendation with social dimensions. In: AAAI, pp. 251–257 (2016)
53. Tang, X., Wang, T., Yang, H., Song, H.: AKUPM: Attention-enhanced knowledge-aware user preference model for recommendation. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1891–1899 (2019)
54. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)
55. Wang, X., He, X., Cao, Y., Liu, M., Chua, T.S.: Kgat: Knowledge graph attention network for recommendation. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 950–958 (2019)
56. Wang, Z., Lin, G., Tan, H., Chen, Q., Liu, X.: CKAN: Collaborative knowledge-aware attentive network for recommender systems. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 219–228 (2020)
57. Wang, Q., Mao, Z., Wang, B., Guo, L.: Knowledge graph embedding: A survey of approaches and applications. IEEE Trans. Knowl. Data Eng. **29**(12), 2724–2743 (2017)
58. Wang, X., Wang, D., Xu, C., He, X., Cao, Y., Chua, T.S.: Explainable reasoning over knowledge graphs for recommendation. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 5329–5336 (2019)
59. Wang, X., Xu, Y., He, X., Cao, Y., Wang, M., Chua, T.S.: Reinforced negative sampling over knowledge graph for recommendation. In: Proceedings of The Web Conference 2020, pp. 99–109 (2020)
60. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: AAAI, pp. 1112–1119 (2014)
61. Wang, H., Zhang, F., Wang, J., Zhao, M., Li, W., Xie, X., Guo, M.: Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, pp. 417–426 (2018)
62. Wang, H., Zhang, F., Xie, X., Guo, M.: DKN: Deep knowledge-aware network for news recommendation. In: Proceedings of the 2018 World Wide Web Conference, pp. 1835–1844 (2018)
63. Wang, H., Zhang, F., Zhang, M., Leskovec, J., Zhao, M., Li, W., Wang, Z.: Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 968–977 (2019)

64. Wang, H., Zhang, F., Zhao, M., Li, W., Xie, X., Guo, M.: Multi-task feature learning for knowledge graph enhanced recommendation. In: The World Wide Web Conference, pp. 2000–2010 (2019)
65. Wang, H., Zhao, M., Xie, X., Li, W.J., Guo, M.Y.: Knowledge graph convolutional networks for recommender systems. In: Proceedings of the 27th International World Wide Web Conference, pp. 3307–3313 (2019)
66. Wu, Q., Jiang, L., Gao, X., Yang, X., Chen, G.: Feature evolution based multi-task learning for collaborative filtering with social trust. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence, pp. 3877–3883 (2019)
67. Wu, L., Sun, P., Fu, Y., Hong, R., Wang, X., Wang, M.: A neural influence diffusion model for social recommendation. In: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 235–244 (2019)
68. Yang, B., Lei, Y., Liu, J., Li, W.: Social collaborative filtering by trust. IEEE Trans. Pattern Anal. Mach. Intell. **39**(8), 1633–1647 (2016)
69. Ying, H., Wu, J., Xu, G., Liu, Y., Liang, T., Zhang, X., Xiong, H.: Time-aware metric embedding with asymmetric projection for successive POI recommendation. World Wide Web **22**(5), 2209–2224 (2019)
70. Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks? In: Advances in Neural Information Processing Systems, pp. 3320–3328 (2014)
71. Yu, X., Ren, X., Gu, Q., Sun, Y., Han, J.: Collaborative filtering with entity similarity regularization in heterogeneous information networks. In: IJCAI (2013)
72. Yu, X., Ren, X., Sun, Y., Gu, Q., Sturt, B., Khandelwal, U., Norick, B., Han, J.: Personalized entity recommendation: A heterogeneous information network approach. In: Proceedings of the 7th ACM International Conference on Web Search and Data Mining, pp. 283–292 (2014)
73. Zhang, Z., Liu, Y., Zhang, Z., Shen, B.: Fused matrix factorization with multi-tag, social and geographical influences for POI recommendation. World Wide Web **22**(3), 1135–1150 (2019)
74. Zhang, H., Shen, F., Liu, W., He, X., Luan, H., Chua, T.S.: Discrete collaborative filtering. In: Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, pp. 325–334 (2016)
75. Zhang, C., Yu, L., Wang, Y., Shah, C., Zhang, X.: Collaborative User Network Embedding For Social Recommender Systems. In: SDM, pp. 381–389 (2017)
76. Zhang, F., Yuan, N.J., Lian, D., Xie, X., Ma, W.Y.: Collaborative knowledge base embedding for recommender systems. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 353–362 (2016)
77. Zhao, H., Yao, Q., Li, J., Song, Y., Lee, D.L.: Meta-graph based recommendation fusion over heterogeneous information networks. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 635–644 (2017)
78. Zheng, X., Luo, Y., Sun, L., Ding, X., Zhang, J.: A novel social network hybrid recommender system based on hypergraph topologic structure. World Wide Web **21**(4), 985–1013 (2018)