Neurocomputing 454 (2021) 417–429

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

# Knowledge Graph enhanced Neural Collaborative Filtering with Residual Recurrent Network

Lei Sang [a,b,c], Min Xu [b,*], Shengsheng Qian [d], Xindong Wu [c,e]

[a] School of Computer Science and Technology, Anhui University, Hefei, 230000, China
[b] Faculty of Engineering and Information Technology, University of Technology Sydney, Sydney 2007, Australia
[c] School of Computer Science and Information Engineering, Hefei University of Technology, Hefei 230009, China
[d] Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
[e] Key Laboratory of Knowledge Engineering with Big Data (Hefei University of Technology), Ministry of Education, Hefei 230009, China

ARTICLE INFO

*Article history:*
Received 12 January 2020
Revised 11 October 2020
Accepted 13 March 2021
Available online 2 April 2021
Communicated by Zidong Wang

*Keywords:*
Recommendation system
Knowledge Graph
Relational Path Embedding
Neural Collaborative Filtering
Residual Recurrent Network

ABSTRACT

Knowledge Graph (KG), which commonly consists of fruitful connected facts about items, presents an unprecedented opportunity to alleviate the sparsity problem in recommender system. However, existing KG based recommendation methods mainly rely on handcrafted meta-path features or simple triple-level entity embedding, which cannot automatically capture entities' long-term relational dependencies for the recommendation. Specially, entity embedding learning is not properly designed to combine user-item interaction information with KG context information. In this paper, a two-channel neural interaction method named **K**nowledge **G**raph enhanced **N**eural **C**ollaborative **F**iltering with **R**esidual **R**ecurrent **N**etwork (**KGNCF-RRN**) is proposed, which leverages both long-term relational dependencies KG context and user-item interaction for recommendation. (1) For the KG context interaction channel, we propose Residual Recurrent Network (RRN) to construct context-based path embedding, which incorporates residual learning into traditional recurrent neural networks (RNNs) to efficiently encode the long-term relational dependencies of KG. The self-attention network is then applied to the path embedding to capture the polysemy of various user interaction behaviours. (2) For the user-item interaction channel, the user and item embeddings are fed into a newly designed two-dimensional interaction map. (3) Finally, above the two-channel neural interaction matrix, we employ a convolutional neural network to learn complex correlations between user and item. Extensive experimental results on three benchmark datasets show that our proposed approach outperforms existing state-of-the-art approaches for knowledge graph based recommendation.



## 1. Introduction

As the amount of data from different platforms grows at an unprecedented rate, there is an enormous demand for the recommendation system to provide relevant information tailored to users' interests or preferences. For example, 11,914 movies are released around the world in 2018.[1] It is impossible for users to watch all available movies to identify their interested ones. As a result, the recommender system is becoming more and more crucial to help users discover personalised content from these ever-growing corpus of data. Due to their extraordinary capability of exploiting collective wisdom and experiences, Collaborative Filtering (CF) algorithms have proven to be the most widely used recommendation technology [37]. Though widely studied in the past, traditional collaborative filtering, however, suffers from severe user-item interaction data sparse issues [8].

Inspired by the recent popularity of deep neural networks, Neural Collaborative Filtering (NCF) is proposed, as a new class of CF methods, cast the traditional MF (Matrix Factorization) algorithm into an overall neural framework [13,12,6,20]. In general, learnable NCF models consist of two key components: 1) embedding, which converts users and items to vectorised embeddings, and 2) interaction modelling, which encodes historical interactions over the user/item embeddings. For example, in [13], the inner product of the MF interaction function is replaced by deep nonlinear networks. And the Euclidean distance metric is used in translation-based CF as the interaction function [34]. However, these methods lack the ability of producing high-quality user/item embeddings

* Corresponding author.
*E-mail address:* Min.Xu@uts.edu.au (M. Xu).

[1] https://www.imdb.com/search/title?year=2018&title_type=feature&.

https://doi.org/10.1016/j.neucom.2021.03.053
0925-2312/

for collaborative filtering. One critical reason is that most existing methods build the embedding function with the descriptive features only (e.g., ID or categories). As a result, when the neural-based interaction function goes deeper to capture complex user-item relationship, these methods tend to overfit and make the sparsity problem even worse.

The rapid explosion of the online social network platforms offer new opportunities for CF. Therefore, researchers propose using auxiliary data in recommender system to enrich the semantics of user or item embedding, such as social networks, attributes, and multimedia data [28,15,18,23,27]. In previous works, these auxiliary information is usually organised in an unstructured manner, and they only available in specific platforms. Recently, Knowledge Graphs (KGs) have attracted increasing attention, which usually consist of fruitful connected facts about items organised in graph structure [22,40,3]. By introducing semantic relatedness among items, KG is a useful tool to help mine items' hidden connections and provide better recommendation results. Extra user-item connectivity context information derived from KG endows recommender systems the ability of reasoning and alleviates sparsity [45]. Taking movie recommendation as an example in Fig. 1, a user *Bob* is connected to the movie *"The Revenant"*, since she likes the movie *"Inception"* by the same actor *"Leonardo DiCaprio"*. Such connectivity helps to reason about unseen user-item interactions (*i.e.,* a potential recommendation) by synthesising information from long paths, which are complementary to sparse user-item interaction data.

There are mainly two kinds of knowledge graph enhanced recommender systems: path-based model and entity embedding based models (1) Path-based models usually use pre-defined meta-path with specific connectivity patterns to capture different semantics similarities between users and items carried in KGs. Meta-path exploits KG in an intuitive way, however, they heavily rely on manually-designed features to capture path semantics, and they fail to automatically uncover and reason on unseen connectivity patterns. More importantly, handcrafted features can hardly exploit all of the possible entity connections, thus limiting the recommendation results' quality. (2) For KG embedding based models, the KG is usually pre-processed by Knowledge Graph Embedding (KGE) methods [42], such as TransE [2] and TransR [19], and then the learned entity embeddings are used to regularise the representations of items. KGE start with the assumption that similar entities are likely to have similar relational roles, where each triplet $(h, r, t)$ in KG is interpreted as $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$, indicates a fact that *head* entity $h$ and *tail* entity $t$ have the relationship $r$. Here we use the boldfaces to denote their embeddings. Their primary focus, therefore, depends on learning from relational triplets of entities. However, triple-level learning has the limitation of low expressiveness. These methods only consider direct relations between entities, rather than the long-term path connectivity. Furthermore, simple entity embedding disregards the semantic relations of entities that are connected by paths, which has been widely used in path based methods.

To address these issues, we aim to find a new data-driven method that does not rely on manually designed meta-path features or simple entity embedding, yet can still obtain the hidden semantics of both entities and paths within KGs for the recommendation. Our main idea is to learn *context-based path embedding* that explicitly exploit long-term dependencies relational context in KG, which characterise a three-way interaction of the form: (user, KG, item) tailored for the recommendation task. Take the KG based movie recommendation, for example, where Bob's preference can be inferred by the relational paths: 1) Bob $\xrightarrow{rate}$ Inception $\xrightarrow{genre}$ Fantasy $\xrightarrow{genre}$ Batman Begins, 2) Bob $\xrightarrow{rate}$ Inception $\xrightarrow{directedby}$ Christopher Nolan $\xrightarrow{direct}$ Batman Begins. Learning the representation of above relational paths can exploit more relational dependencies than simple entity triplets, while still maintain the local relational information of entities. Moreover, without time-consuming manual meta-path design, path embedding can still exploit multi-hop relation among entities. And the low-dimension path embedding can also facilitate the downstream process with more flexibility. This example also highlights that different paths connecting a same entity pair often carry relations of different semantics. User behaviours have the polysemy property, since an action to an item may have a variety of meanings in different contextual scenarios. Typically, they are of different importance in characterising user tastes over items, i.e., certain paths can better describe user preferences than the others. In the example, Bob's preference over Batman Begins may be driven more by his interest in the genre than by his favour for the director. To fully exploit paths in KGs for recommendation, it requires to capture not only the semantics of different paths but also their distinctive saliency in describing user preferences towards items.1.

Towards this end, we propose a two-channel neural interaction model by explicitly incorporating path based KG context into the sparse user-item interaction. (1) To construct the context-based path interaction, instead of using traditional Recurrent Neural Networks (RNNs), we propose *Residual Recurrent Network* (RRN), which is optimised for sequence modelling in KG. That is because RNNs predict the next element of the sequence only based on the current input and the previous hidden state, which is inappropriate for KG path modelling with the triplet structure. To overcome this weakness, the proposed RRN enable the output hidden states of relations to learn a *residual* [11] connection from their direct head entities when inferring tail entities in the relational sequence. A self-attention network is then applied to the context-based path embedding to capture the polysemy of user interaction behaviours. (2) To explicitly exploit the pairwise interaction between user and item, we propose to use outer product above the embedding layer. Outer product operation generates a two-dimensional interaction map that is more expressive and semantically plausible than simple concatenation or element-wise product. (3) Above these two interaction matrix channel, we propose to adopt a convolutional neural network (CNN) to learn the high-order correlations between users and items. Finally, a joint learning framework is designed to train the overall neural framework in an end-to-end manner, which makes the KG path embedding part and the neural collaborative recommendation (NCF) part to enhance the training process of each other mutually.

The main contributions of this paper are as follows:

1. We are the first to combine KG structure information with the neural collaborative recommendation in an end-to-end neural style.
2. We propose a new recommendation framework KGNCF-RRN, which capture both of 1) the KG's long-term dependencies context encoded in path embedding and 2) complex matching interaction of user-item with a two-channel convolutional NCF.
3. We conduct empirical studies on three large-scale real-world dataset. Extensive experiment results demonstrate the state-of-the-art performance of KGNCF-RRN and its effectiveness in improving the embedding and interaction quality for the final prediction.

## 2. Preliminary

Before introducing our proposed approach, we first introduce the notation and the definition knowledge graph enhanced

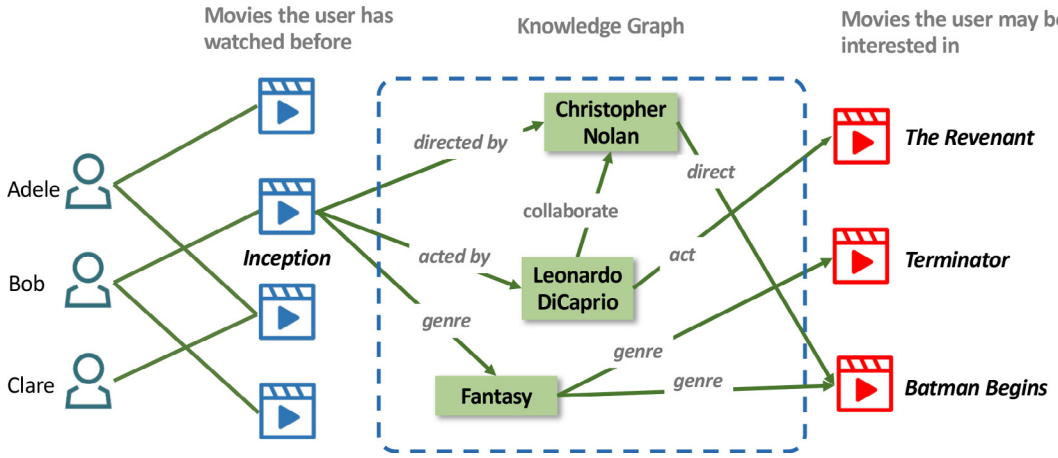


**Fig. 1.** Knowledge graph enhanced movie recommender systems. The knowledge graph in our system can provides extra user-item connectivity information, which are useful for alleviating the sparsity problem of target recommendation task.

**Table 1**
Notations and explanations.

| Notations | Description |
|---|---|
| $y_{ui} \in Y$ | Users' implicit feedback |
| $\hat{y}_{ui}$ | Predicted engaging probability |
| $\mathcal{G} = (\mathcal{E}, \mathcal{R})$ | Knowledge graph |
| $(h, r, t)$ | A knowledge triple (head, relation, tail) |
| $\mathcal{E} = \{e_1, e_2, \cdots\}$ | Set of entities |
| $\mathcal{R} = \{r_1, r_2, \cdots\}$ | Set of relations |
| $p_s \in \mathcal{P}(u, i)$ | Knowledge paths for user-item pair |
| $p_s = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$ | each path sequence |
| S | the dimension of hidden space; number of relational paths |
| $\mathbf{h}'_s$ | path embedding for $p_s$ |
| $\mathbf{k}_u \in \mathbb{R}^S$ | latent vector for user $u$ |
| $\mathbf{j}_i \in \mathbb{R}^S$ | latent vector for item $i$ |
| $\mathbf{p}_i \in \mathbb{R}^S$ | item embedding $i$ |
| $\mathbf{q}_j \in \mathbb{R}^S$ | item embedding $j$ in user history |

recommendation task. Then we shortly recapitulate the standard neural collaborative filtering, highlighting its limitations for dealing with this task.

In the study of KG-enhanced recommendation, we have a target recommendation domain with user-item historical interaction data and an auxiliary knowledge graph domain data. For the data in the target recommendation task, let $\mathcal{U} = \{u_1, u_2, \ldots, u_M\}$ denotes a set of users, and $\mathcal{I} = \{i_1, i_2, \ldots, i_N\}$ denotes a set of items, where $M$ and $N$ are used to denote the number of users and items, respectively. We usually use a bipartite graph to represent the user-item historical interaction records. Thus, we construct a matrix of user-item interaction $Y \in \mathbb{R}^{M \times N}$ based on the user-item interaction as following:

$$y_{ui} = \begin{cases} 1 & \text{if user } u \text{ has interacted with item } i; \\ 0 & \text{otherwise.} \end{cases}$$

Here an observed interaction ($y_{ui} = 1$) imply that user $u$ interact with item $i$, e.g., users browsed a piece of news or user purchased a product; however, it can only indirectly reflect users' preference. Similarly, $y_{ui} = 0$ does not naturally mean user $u$ has no interest in item $i$, which could also be that user $u$ is not noticing the item $i$ since the overwhelming amount of items available in a system. The unobserved entries of user-item interaction matrix usually only take a tiny proportion of the whole entries, which brings significant challenges in learning from these sparse interaction data [4].

The auxiliary knowledge Graph (KG) is a directed graph, whose nodes are entities $\mathcal{E}$ (such as real-world objects, events, concepts) and edges $\mathcal{R}$ indicates relations among entities. Formally, we define KG as a set of triplet in form of $\mathcal{G} = \{(h, r, t) | h, r \in \mathcal{E}, r \in \mathcal{R}\}$, where each triplet $(h, r, t)$ represents there is a fact that the *head* entity $h$ and *tail* entity $t$ have the specific type of relationship $r$.

Within $\mathcal{G}$, we formally define the relational path from the user $u$ to the item $i$ as a sequence of entities and relations: $p = [e_1 \xrightarrow{r_1} e_2 \xrightarrow{r_2} \cdots \xrightarrow{r_{L-1}} e_L]$, where $e_1 \in \mathcal{U}, e_L \in \mathcal{I}; (e_l, r_l, e_{l+1})$ is a triplet in the path, and $L$ denotes the number of entities in the path. Assumes a user-item pair $(u, i)$ can be connected by paths of different lengths, i.e., $\mathcal{P}(u, i) = \{p_1, p_2, \ldots, p_S\}$. Note that $S$ is dynamic, as different user-item pairs may be connected with different number of paths. Such connectivity helps to reason about unseen user-item interactions (*i.e.,* a potential recommendation) by synthesising information from long paths, which are complementary to sparse user-item interaction data. Therefore, from the view of reasoning, we expect our method to model the sequential dependencies of entities and sophisticated relations of a path connecting a user-item pair.

**Task Definition**: Given the user-item interaction matrix $Y$, knowledge graph $\mathcal{G}$ and a set of paths $\mathcal{P}(u, i) = \{p_1, p_2, \cdots, p_S\}$ connecting user $u$ and item $i$, we aim to discover users' potential interests for the unobserved feedbacks with the help of auxiliary KG data:

$$\hat{y}_{ui} = f_\Theta(u, i | \mathcal{P}(u, i), \mathcal{G}, Y) \tag{1}$$

where $f$ indicates the prediction model with parameters $\Theta$, and $\hat{y}_{ui}$ denotes the predicted likeliness score for the unknown user-item interaction.

### 2.1. Neural collaborative filtering framework

Collaborative filtering (CF) is the fundamental method in personalised recommendation systems, of which model based Matrix Factorisation (MF) is one of the simplest yet effective implementations [17,49]. MF characterises both user and item with real-valued low-dimensional latent vectors, which infer a recommendation based on the high correspondence between item and user with the inner product:

$$\hat{y}_{ui} = f_{ui}(u, i | \mathbf{k}_u, \mathbf{j}_i) = Trans \mathbf{k}_u \mathbf{j}_i = \sum_{s=1}^{S} k_{us} j_{is} \tag{2}$$

where $\mathbf{k}_u \in \mathbb{R}^S$ and $\mathbf{j}_i \in \mathbb{R}^S$ indicate the hidden vector for user $u$ and item $i$, respectively. And $S$ is the dimension of the hidden space. Despite its effectiveness, we note that using the simple and fixed inner product can hardly estimate complex user-item interactions, which will limit the expressiveness of MF. It is straightforward to understand the inner product function as first applying the element-wise product on latent vectors $\mathbf{k}_u$ and $\mathbf{j}_i$, followed by linearly combining each element with the same weight. And each dimension of latent factors is treated independently from all others. As such, MF with one hidden layer only can only be a linear method [44].

To overcome the limitation of linear MF model, it is natural to model the user-item interaction in neural network style. Thus, Neural Collaborative Filtering (NCF) [13] with a two-way neural network, as shown in Fig. 2. Let $M_u$ and $M_i$ denote the feature information, or simple one-hot representation of user $u$ and item $i$. We define the prediction function as:

$$\hat{y}_{ui} = f(K^\top M_u, J^\top M_i | \mathscr{U}, \mathscr{I}, \theta) \tag{3}$$

where $K \in \mathbb{R}^{M\times S}$ and $J \in \mathbb{R}^{N\times S}$ indicates the embedding matrix for use/item features; $f(\cdot)$ denotes the non-linear multilayer perceptron (MLP) with parameters $\theta$. Traditional MF-based recommendation method is actually the simple case of NCF, while the MLP hidden layer of NCF can model more complex user-item interaction.

Although directly using a neural network to learn the matching score brings excellent flexibility to the model, there is no practical guarantee that the dimension correlations can be effectively captured with current optimisation techniques, especially with the complex user and item interaction [12]. The case can be even worse when we take the auxiliary attributes information (such as knowledge graph, and multimedia content) into account. To overcome shortages of the above issues and further improve the performance of CF methods, we incorporate long-term relational dependencies KG context into user-item interaction for recommendation under the proposed recommendation **KGNCF-RRN** framework.

## 3. The proposed approach

The overall framework of KGNCF-RRN is presented in Fig. 3, which have two major parts: (1) A residual recurrent network (RRN) for path embedding (2) a two-channel convolutional NCF that model both of the KG's long-term dependencies encoded in path embedding and complex matching interaction of user-item. Given user-item interaction and KG data, our aim to capture the fruitful heterogeneous context information hidden in the KG to supplement sparse user-item interaction prediction.

### 3.1. Residual Recurrent Network (RRN) for rational path embedding

We describe the proposed residual recurrent networks model for KG path embedding. The directed path between user-item pair within KG can be seen as a sequence, where elements in the sequence are the entities in the path. We try to map each entity in path $p_s = [\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_{L-1}, \mathbf{e}_L]$ into a low-dimensional distributed embedding, and then preserving the latent meaning of entity sequence into a path embedding. However, only considering the entity in path can not fully exploit the hidden subtle semantic difference between neighbouring entities. Because in most realistic scenarios, the same entity–entity pairs with different connected relations may convey different meaning. And these differences can be seen as the user's diverse preference for choosing different items. Take the two triplets (*Orson Welles, IsDirectorOf, Citizen Kane*) and (*Orson Welles, IsActorOf, Citizen Kane*) appears in two paths respectively as an example to denote a user's preferences. If we do not consider the relation between entities, these two path will have the same embedding representation due to the same entity pair (*Orson Welles, Citizen Kane*), leaving out the possibility that the user may have more interest in movies directed by *Orson Welles*, rather than that acted by *Orson Welles*. Thus, it is necessary to explicitly include the semantics of relations into the path embedding learning. To accomplish this, each relation $r_l$ in $p_s$ is also embedded as a low-dimensional vector $\mathbf{r}_l$. As a result, we obtain a set of embeddings for path $p_s = [\mathbf{e}_1, \mathbf{r}_1, \mathbf{e}_2, \mathbf{r}_2, \cdots, \mathbf{e}_{L-1}, \mathbf{r}_{L-1}, \mathbf{e}_L]$, where each element denotes an entity or a relation. For simplicity, we use $p_s = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$ denote the same path, where $T$ is the length of the path sequence by considering both entities and relations: $T = 2 * L - 1$.

#### 3.1.1. RNN

After modelling the relational path between user-item pair as a sequence, it seemed fairly natural for us to employ the RNN model to learn the path embedding. This is primarily because RNN has the ability to encode variable length sequences and preserve the semantics of the entire path between user-item pair. Here we choose GRU as our basic sequential model, which is an RNN variant that tries to address gradient vanishing and exploding problem in sequence. Given a relational path $p_s = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$ as input, GRUs sequentially read in elements in this sequence and output a hidden state at each time step. It has an *update* gate $\boldsymbol{z}_t$ and a *reset* gate $\boldsymbol{r}_t$ to control the flow of information. The formulas are:

$$\begin{aligned}
\boldsymbol{z}_t &= \sigma(\boldsymbol{U}_z \mathbf{x}_t + \boldsymbol{W}_z \boldsymbol{h}_{t-1} + \boldsymbol{b}_z) \\
\boldsymbol{r}_t &= \sigma(\boldsymbol{U}_r \mathbf{x}_t + \boldsymbol{W}_r \boldsymbol{h}_{t-1} + \boldsymbol{b}_r) \\
\tilde{\boldsymbol{h}}_t &= \tanh(\boldsymbol{U}_c \mathbf{x}_t + \boldsymbol{W}_c (\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1}) + \boldsymbol{b}_c) \\
\boldsymbol{h}_t &= (1 - \boldsymbol{z}^t) \odot \boldsymbol{h}_{t-1} + \boldsymbol{z}_t \odot \tilde{\boldsymbol{h}}_t
\end{aligned} \tag{4}$$

where $\mathbf{x}_t \in p_s$ is the input vector of path sequence and $t$ is the time step, the *logistic* function $\sigma(x) = 1/(1 + \exp(-x))$ is used to do non-linear projection, $\odot$ is the element-wise product between two vectors. $\tilde{\boldsymbol{h}}_t$ is the candidate state activated by element-wise $\tanh(x)$. The output $\boldsymbol{h}_t$ is the current hidden state. The last hidden state $\boldsymbol{h}_T$ is usually regarded as the sequence's representation, where $T$ is the length of the relational path.

RNN are capable of processing variable length input with a few parameters and have widely used in many sequential modelling tasks. However, when applying RNN to model relational paths in KG, there may still face following two challenges: (1) There are two different types of elements in a relational path: "entity" and "relation", and they always appear alternately. However, the traditional RNN do not consider the type of each element and treat them equally, which may lead to the loss of crucial dependency information among different type of elements. (2) Triple is the basic structure of all the relational paths. However, RNNs ignore this important structure information. Specifically, given a KG relational sequence $\cdots \rightarrow \mathbf{x}_{t-1} \rightarrow \mathbf{x}_t \rightarrow \mathbf{x}_{t+1} \rightarrow \cdots$, where $(\mathbf{x}_{t-1}, \mathbf{x}_t, \mathbf{x}_{t+1})$ forms a triple. To predict $\mathbf{x}_{t+1}$, RNN would combine the hidden state $\mathbf{h}_{t-1}$ and the current input $\mathbf{x}_t$, where $\mathbf{h}_{t-1}$ is a mix of the information of all the previous elements $\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}$. In Knowledge graph embedding theory, the semantic meaning of tail entity $\mathbf{x}_{t+1}$ is determined primarily by head entity $\mathbf{x}_{t-1}$ and relation $\mathbf{x}_t$ in the triple as: $\mathbf{x}_{t-1} + \mathbf{x}_t \approx \mathbf{x}_{t+1}$, and has little to do with the sequence before $\mathbf{x}_{t-1}$. Thus, we should put more emphasised on the information of $\mathbf{x}_{t-1}$ and $\mathbf{x}_t$ in the current triple to predict the following tail entity $\mathbf{x}_{t+1}$[9].

#### 3.1.2. Residual Recurrent Network (RRN)

To address the problem of traditional RNNs and better model the triple structure in the rational path, a simple yet effective residual mechanism [11] is proposed to reform RNN, which is called

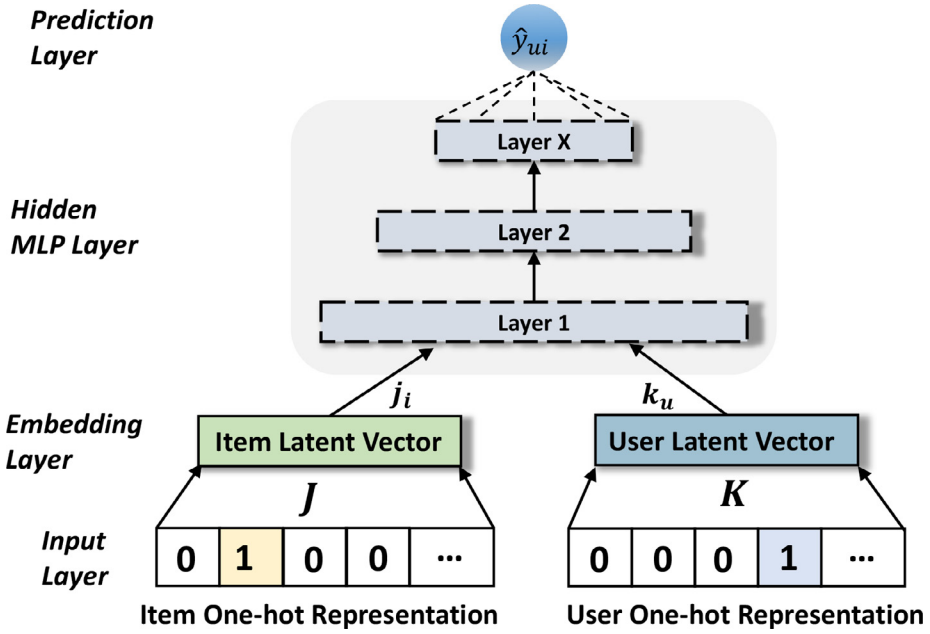


**Fig. 2.** Neural Collaborative Filtering (NCF): MF as a shallow neural network model.

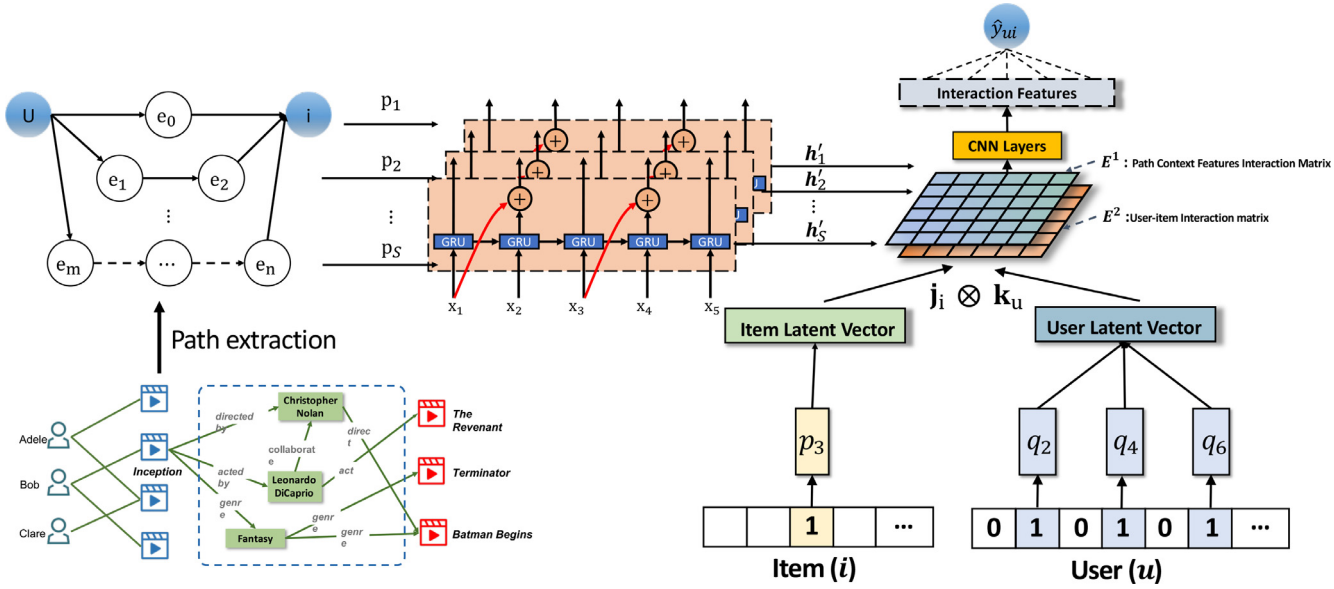


**Fig. 3.** Illustration of Knowledge Graph Enhanced Neural Collaborative Filtering with Residual Recurrent Network (KGNCF-RRN). (1) For the KG context interaction channel, the proposing of Residual Recurrent Network (RRN) is used to construct context-based path embedding. The self-attention network is then applied to the path embedding to capture the polysemy of various user interaction behaviours. (2) For the user-item interaction channel, the user and item embeddings are fed into a newly designed two-dimensional interaction map. (3) Finally, above the two-channel neural interaction matrix, we employ a convolutional neural network to learn complex correlations between user and item.

Residual Recurrent Network (RRN). For each of the triples in the relational path, the residual mechanism try to add another shortcut connections from the head entity to tail entity, where the head entity is straightly used to predict the target tail entity. Thus, an element in a relational path with type "entity" can not only predict its following "relation", but also directly contribute to predicting its target tail entity. Fig. 4 illustrates an RRN example.

For a relational path $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$, the residual operation for RRN is formulated as follows:

$$\mathbf{h}'_t = \begin{cases} \mathbf{h}_t & \mathbf{x}_t \in \mathscr{E} \\ \mathbf{W}_1\mathbf{h}_t + \mathbf{W}_2\mathbf{x}_{t-1} & \mathbf{x}_t \in \mathscr{R} \end{cases}, \tag{5}$$

where $\mathbf{h}'_t$ denotes the output hidden state of the RRN at time step $t$, and $\mathbf{h}_t$ denotes the corresponding GRU output. $\mathbf{W}_1$, $\mathbf{W}_2$ are the weight matrices, and we share their parameters at different time steps. In this paper, we choose the weighted sum for the residual operation, but other combination methods (such as concatenation and mean operation) can be employed as well.

Intuitively, RRN explicitly distinguishes entities and relations, and allows head entities to skip their relations for directly participating in target tail entity predication. Different from ResNet [11], which were proposed to help train very deep networks, RRN employ residual learning on "shallow" networks. This kind of skipping/shortcut connections in RRN do not link the previous input to

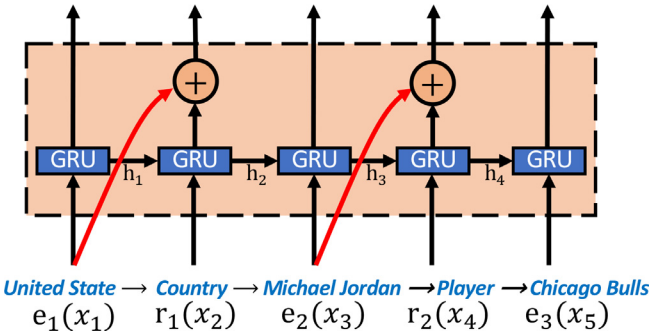


**Fig. 4.** Example of Residual Recurrent Network (RRN). For each of the triples in the relational path, the residual mechanism try to add another shortcut connections from the source entity to target entity, where the source entity is straightly used to predict the target entity. Thus, the crucial triplet structure information is also encoded in the path embedding.

the very deep layers, but only concentrate on each specific triple structure in a KG relational sequence. The similar technique was originally applied in knowledge graph embedding as skipping learning [9]. The shortcut connections in Eq. (5) introduce just a few extra parameters and will not increase computation complexity. Besides the practical characteristic, this feature can also make it possible to fairly compare plain/residual networks, given they have a similar number of parameters and computational cost.

#### 3.1.3. Noise-contrastive estimation with type

By learning parameter to predict the next element, we can optimise each element in the path. However, the extracted relational paths contain a large amount of candidate entities or relations, which makes the computation of prediction loss is time-consuming. One idea to solve this problem is: instead of summing over all the training samples, just pick a few. These chosen non-target elements are called Negative Samples. Specifically, we use the noise-contrastive estimation (NCE) [10] to evaluate each output of RRN, which only requires a small number of negative samples to approximate the integral distribution. Finally, the loss of RRN is defined as follows:

$$\begin{aligned}\mathscr{L} = & -\sum_{t=1}^{T-1}(\log\sigma(\mathbf{h}'_t\cdot\mathbf{c}_t) \\ & +\sum_{j=1}^{k}\mathbb{E}_{\tilde{c}_j\sim Q(\tilde{c})}\left[\log\sigma(-\mathbf{h}'_t\cdot\tilde{\mathbf{c}}_j)\right])\end{aligned} \tag{6}$$

where $\mathbf{c}_t$ is the target at time step $t$, $\sigma(\cdot)$ is the sigmoid function, and $k$ is the number of negative samples. A negative example $\tilde{c}_j$ is drawn from the noise probability distribution: $Q(\tilde{c}) \propto q(\tilde{c})^{\frac{3}{4}}$, where $q(\tilde{c})$ is the frequency of $\tilde{c}$ appearing in knowledge graph [9].

### 3.2. Multi-channel interaction modelling

A user-item entity pair usually has a set of paths connecting them in a KG, which provide fruitful context information complement to sparsity user-item interaction. In this section, we construct a two-channel matrix to fully exploit theses context information. (1) A path context interaction channel is built to model the high-order path embeddings interaction from KG. (2) For the user-item interaction channel, the user and item embeddings are fed into a newly designed two-dimensional interaction map.

#### 3.2.1. Path context interaction channel with self-attention

For each relational path in $\mathscr{P}(u,i) = \{p_1, p_2, \cdots, p_S\}$, the RRN hidden state output for the last time step can be seen as their corresponding path embedding: $(\mathbf{h}'_1, \mathbf{h}'_2, \cdots, \mathbf{h}'_S)$. Once the path features encoded to low-dimensional space, we try to model high-order combinatorial path features in this space. As there are $S$ paths linking user $u$ and item $i$ in $\mathscr{P}(u,i)$, different paths may play different roles in modelling the relations between them. And these path features may jointly contribute to the final prediction. Thus, we need to decide which features should be combined to form meaningful high-order features for prediction. However, the combination of different features (a.k.a. cross features) is very time-consuming to manually designed by domain experts. More than that, due to the combinatorial explosion problem, it is impossible to enumerate all the combinations. Therefore, we need to find a way that can automatically exploit the meaningful combinations of different path features. In this paper, self-attention mechanism [35] is applied to solve this problem instead of hand-craft feature engineering. Specifically, a novel interacting layer is proposed to model the interactions between different path embedding features. The interacting layer considers pairwise path feature interactions, and can automatically recognise interrelated features to construct practical higher-order features via the self-attention mechanism.

Self-attention network (SAN) has recently achieved great success in modelling complex interrelationships. For example, it is widely applied to model arbitrary word interaction in machine translation [35] or text embedding [31], and also achieves excellent performances in capturing node similarities in graph neural network [36]. Here we extend Self-attention mechanism to capture the interactions between different path embedding features. Take the path embedding $(\mathbf{h}'_1, \mathbf{h}'_2, \cdots, \mathbf{h}'_S)$ obtained from RRN as input of the self-attention model, the self-attention matrix is defined as $E^1_{ij}$:

$$\begin{aligned} f(\mathbf{h}'_i, \mathbf{h}'_j) &= W\sigma(W_1\mathbf{h}'_i + W_2\mathbf{h}'_j) \\ E^1_{i,j} &= \frac{\exp(f(\mathbf{h}'_i,\mathbf{h}'_j))}{\sum_{j=1}^{S}\exp(f(\mathbf{h}'_i,\mathbf{h}'_j))} \end{aligned} \tag{7}$$

where $f(\mathbf{h}'_i,\ \mathbf{h}'_j)$ is an attention function, which defines the similarity between the path embedding feature $i$ and $j$. Generally, we can also use a neural network or an inner product to represent this function. In this work, we adopt a weighted sum due to its simplicity and effectiveness. $W_1$, $W_2$ denotes the transformation matrices that mapping the original embedding space $\mathbb{R}^d$ into a new space $\mathbb{R}^{d'}$.

Here $E^1_{ij} \in \mathbf{E}^1$ is the interaction coefficient of path embedding feature $i$ and its relevant features, it represents a new combinatorial feature learned by our method. As pointed out by Sun et al. [32], short length paths are good enough to model the user-item relation context, whereas too long paths may even deteriorate the performance due to the extra noise introduced. As the number of paths grows exponentially w.r.t. the length of path, where millions of interlinks will be generated. Thus, we truncate all paths at a certain length (each with length up to six) and disregarding remote connections, which are sufficient to model the connectivity between a user-item pair. The average path length for three datasets(MovieLens-1M,Book-Crossing and Last.FM) are 5.12, 5.08 and 5.05 respectively. We filter out the first $S$ number of shortest paths, so the dimension of the $\mathbf{E}^1$ is $S \times S$. We extract qualified paths with different semantics that connect entity pairs (i.e., user-item) in an automatic fashion, instead of handcrafted features (e.g., meta paths) from the KG.

#### 3.2.2. User-item outer product interaction channel

Besides the KG context, to capture the complex interaction between user and item embeddings, we use an outer product operation over the user and item embedding layer to generate a two-dimensional interaction map. The interaction map is rather suitable for the CF task, since it not only subsumes the interaction

signal used in MF (its diagonal elements correspond to the intermediate results of inner product), but also learns possible complex dimension correlations with CNN layers. Thus, the auxiliary knowledge information and complex nonlinear interaction are modelled in the proposed unified architecture. To obtain the complex correlation matrix, we apply the outer product on user latent vector $\mathbf{k}_u$ and item latent vector $\mathbf{j}_i$. This correlation matrix is term as the *interaction map* with the dimension of $S \times S$:

$$\mathbf{E}^2 = \mathbf{k}_u \otimes \mathbf{j}_i = \mathbf{k}_u \mathbf{j}_i^\top \tag{8}$$

where $\mathbf{E}^2$ is a $S \times S$ matrix, in which each element is evaluated as: $e_{s_1,s_2} \in \mathbf{E}^2 = \mathbf{k}_{u,s_1} \mathbf{j}_{i,s_2}$. Here the $S$ is set to 64.

The idea of interaction map is a critical design to ensure the effectiveness of KGNCF-RRN for the recommendation. Compared to prior work [13,53], we argue that outer product can benefit the recommendation from three aspects: 1) interaction map can encode more dimension correlation information than matrix factorisation (MF), due to the fact that MF considers only diagonal elements in the interaction map; 2) interaction map with rich semantics facilitate the following non-linear layers to generalise well on sparse data. and 3) the 2D matrix format of interaction map makes it feasible to learn the interaction function with the effective CNN, which is known to generalise better and is more easily to go deep than the fully connected MLP.

### 3.3. Convolutional NCF

Two-channel interaction matrix $\mathbf{E} = (\mathbf{E}^1 : \mathbf{E}^2)$ encode the information of interaction from three aspects: the involved user, the involved item and the corresponding path based KG context. Above the two-channel interaction map is a stack of hidden layers, which targets at extracting useful signal from these two-channel interaction matrix. It is subjected to design and can be abstracted as $\mathbf{g} = f_\Theta(\mathbf{E})$, where $f_\Theta$ denotes the model of hidden layers that has parameters $\Theta$, and $\mathbf{g}$ is the output vector to be used for the final prediction. Here we choose convolutional networks as the hidden layer to implement $\mathbf{g}$, which stacks layers in a locally connected manner and utilises much fewer parameters than MLP. The similar technique was originally applied in the recommendation for personalised ranking [12]. This allows us to build deeper models than MLP easily, and benefits the learning of high-order correlations among embedding dimensions.

Fig. 5 shows Hidden CNN layers model. After getting the two-channel interaction matrix $\mathbf{E}$, we apply 32 kernels of size $2 \times 2 \times 2$ filters to the input interaction matrix (with size $2 \times 64 \times 64$). Since the stride size is set to 2, the size of feature map $c$ in hidden layer $l$ ($\mathbf{E}_{lc}$) is half of its previous layer $l-1$. Thus the size of feature maps in the first hidden layer is $32 \times 32 \times 32$. Following the similar convolution operation, we can get the feature maps for the following layers. The output of the 6-th layer is a tensor of dimension $1 \times 1 \times 32$, which can be seen as a vector and is projected to the final prediction score. We use the rectifier unit as the activation function, a common choice in CNN to build deep models. Despite that MLP is theoretically guaranteed to have a strong representation ability, the main drawback is the large number of parameters. And MLP also needs to be carefully tuned on the regularisation of each layer to ensure good generalisation performance. Note that convolution filter can be seen as the "locally connected weight matrix" for a layer, since it is shared in generating all entries of the feature maps of the layer. This significantly reduces the number of parameters of a convolutional layer compared to that of a fully connected layer[12].

The prediction layer takes in the 6-th layer vector $\mathbf{g}$ and outputs the prediction score as: $\hat{y}_{ui} = \mathbf{w}^T \mathbf{g}$, where vector $\mathbf{w}$ re-weights the interaction signal in $\mathbf{g}$.

### 3.4. Learning algorithm

The complete loss function of our model is as follows:

$$\begin{aligned} min_\Theta \mathscr{L} = & \sum_{(u,i)} \in \mathscr{D}\mathscr{J}(\hat{y}_{ui}, y_{ui}) + \\ & \alpha[-\sum_{t=1}^{T-1}(\log \sigma(\mathbf{h}'_t \cdot \mathbf{c}_t) \\ & + \sum_{j=1}^{k} \mathbb{E}_{\tilde{c}_j \sim Q(\tilde{c})}[\log \sigma(-\mathbf{h}\prime_t \cdot \tilde{\mathbf{c}}_j)])] \end{aligned} \tag{9}$$

where the first part models the prediction loss (cross-entropy loss), and the second part captures the KG embedding loss as defined in Eq. (6). $\alpha$ denotes the balance parameter. In our model, all model parameters $\Theta$ are trained in a joint manner. Particularly, joint training optimises the parameters in the KG path embedding part and the neural collaborative filtering part simultaneously at the training procedure. In this way, the two parts could mutually influence each other and can learn a more general representation for final prediction.

In our model, all model parameters $\Theta$ are trained in a joint manner. Particularly, joint training optimises the parameters in the KG path embedding part and the neural collaborative filtering part simultaneously at the training procedure. An alternative solution is train the KG Path embedding parameter first and then fine-tune NCF-CNN from the rating information with the remaining parameters a fixed path embedding parameter[47]. In fact, the second approach works as the pretraining and fine tuning of the deep learning. We call this alternative approach as loosely training. In this loosely approach, the rating prediction process entirely relies on the path embedding part while the path embedding part could not benefit from rating information.

## 4. Experiment

### 4.1. Datasets

We use the following three datasets in our experiment for movie, book and music recommendation, respectively:

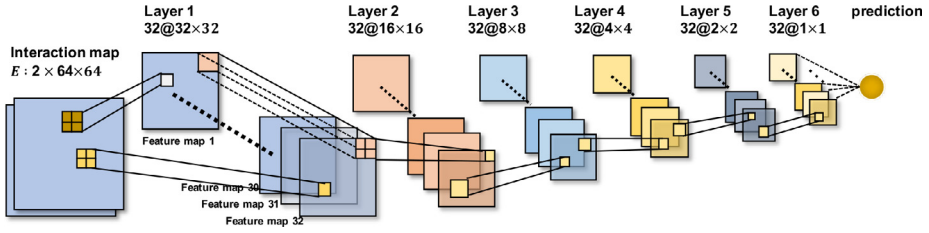


**Fig. 5.** Hidden CNN layers with embedding size 64.

- **MovieLens-1M**[2] dataset contains almost 1 million ratings (ranging from 1 to 5) on the MovieLens website, which is a widely used benchmark dataset in movie recommendations.
- **Book-Crossing**[3] dataset contains 1 million explicit ratings (ranging from 0 to 10) of books in the Book-Crossing community.
- **Last.FM**[4] dataset contains music artist listening information from a set of 2 thousand users at Last.fm online music system. Last.FM provides the listening count as the weight for each user-item interaction.

Following most existing item recommendation that designed to explore implicit feedback, we transform explicit feedback of all three datasets into implicit one. Specially, each implicit feedback is marked as 1, which denotes that a positive rate, while we also randomly sample negative examples from the unknown items. Following previous efforts [40], the threshold of positive rating for MovieLens1M is set to 4, while all ratings in Book-Crossing and Last.FM are set to positive sample due to their sparsity. To construct the knowledge graph for a specific task from DBPedia,[5] we retrieve related triplet facts from DBpedia using SPARQL queries. For simplicity, items with no matched or multiple matched entities are excluded. Besides, we only consider those triplets that are directly related to the entities with the mapped items, no matter which role (*i.e.,* head or tail) the entity serves as.

### 4.2. Baselines

We compare the proposed KGNCF-RRN with two kinds of representative recommendation methods. One is CF-based methods, which only leverage implicit user-item interaction information. Another is the KG-based methods, which incorporate extra knowledge graph information into the user-item interaction.

- **NCF** [13] is a neural network architectures for collaborative filtering. It replaces the inner product with a neural architecture to learn an arbitrary matching function from data.
- **LibFM** [24] is a commonly used feature-based factorisation model for Click Through Rate (CTR) prediction. Only user ID and item ID are concatenated together as the input for LibFM.
- **LibFM + TransE**. We concatenate the raw features of users and items, as well as the corresponding averaged entity embeddings learned from TransE [2] as input for LibFM. The dimension of TransE is set to 32.
- **CKE** [52] is a typical embedding-based recommendation methods, which exploit items' semantic embedding from structural, textual and visual content to assist CF prediction. For a fair comparison, we implement CKE with only additional structural knowledge information. The dimension of entity embeddings is set to 32.
- **PER** [51] regards the KG as a special case of heterogeneous information networks (HIN) and exploits meta-path based representation to model the rich connectivity between users and items. Specially, we manually design "user-item-attribute-item" meta-path as similarity feature.
- **Wide&Deep** [5] is a general deep model for recommendation combining a (wide) linear channel with a (deep) nonlinear channel. Similar to LibFM + TransE, we use the embeddings of users, items, and entities to feed Wide&Deep. The input for Wide&Deep is the same as in LibFM + TransE. The dimension of user, item, and entity is 64, and we use a two-layer deep channel with a dimension of 100 and 50 as well as a wide channel.
- **RippleNet** [38] is a knowledge graph embedding-based method that propagates and aggregates user potential preference on the KG for the recommendation.
- **KGCN** [41] extends non-spectral GCN approaches to the knowledge graph by aggregating neighbourhood information, which captures inter-item relatedness by mining their associated attributes on the KG.
- **KGAT** [43] propose a attentive embedding propagation layer, which adaptively propagates the embeddings from a node's neighbors to update the node's representation.

Besides, to examine the effectiveness of the proposed method, we prepare three simplified variants of our method. One is NCF-CNN, which only model the single user-item outer product interaction channel $E^2$ with CNN, and does not incorporate the knowledge graph for the recommendation. KGNCF-RNN is another variant that using plain RNN to model the dependencies relational path in knowledge graph for the recommendation without using the residual mechanism. To investigate the effectiveness of self-attention, we use meaning pooling layer over paths instead of self-attention. The output of meaning pooling layer is then concatenated with final layer of user-item interaction for prediction. This variant is named KGNCF-polling.

### 4.3. Experiment setup

In KGNCF-RRN, we set the dimension of user and item latent vector as 64, and the number of CNN hidden layers is set as 6. The dropout rate is set as $\rho = 0.2$ for default, which is determined by optimising *AUC* on a validation set. The ratio of training, validation and test for each dataset is set as $6:2:2$. After random choosing a set of data (such as test subset), we will filter out this test subset from the rating list. And then we make the next random selection from the rest of ratings. Thus, different partition (training, validation, test) do not overlap with each other. And we choose the subset all at once, not choose rating one by one. In this way, each rating will only be chosen once. We repeat each experiment for 10 times, and report the average performance. For the other comparison methods, we optimise their parameters in the validation set. We apply the trained model to each piece of interactions in the test set and output the predicted click probability. Besides, we evaluate our method in two experimental scenarios: (1) In the click-through rate (CTR) prediction, we apply the trained model to each piece of interactions in the test set and output the predicted click probability. Average *AUC* and *F*1 on test sets are used as evaluation criterions for CTR prediction. (2) In the top-*K* recommendation, we use the trained model to determine the *K* items with the highest predicted click probability, and choose *NDCG@K* to evaluate the recommended sets.

### 4.4. Overall performance

#### 4.4.1. Comparison with baselines

The results of all methods in CTR prediction and top-*K* recommendation are presented in Table 2 and Fig. 6, respectively.

- In all cases, our proposed KGNCF-RRN outperform all the baseline methods on all the datasets. In particular, KGNCF-RRN improves over the strongest baselines *w.r.t* AUC by 4.7%, 5.5%, and 4.8% in MovieLens-1 M, Book-Crossing, and Last-FM, respectively. By leveraging residual recurrent networks(RRN) for path embedding, KGNCF-RRN is capable of exploring the

[2] https://grouplens.org/datasets/movielens/1m/.
[3] http://www2.informatik.uni-freiburg.de/cziegler/BX/.
[4] https://grouplens.org/datasets/hetrec-2011/.
[5] https://wiki.dbpedia.org/.

**Table 2**
The results of *AUC* and *F1* in CTR prediction.

| Model | MovieLens-1M | | Book-Crossing | | Last.FM | |
|---|---|---|---|---|---|---|
| | AUC | F1 | AUC | F1 | AUC | F1 |
| NCF | 0.844 (−10.0%) | 0.773 (−8.1%) | 0.642 (−12.8%) | 0.597 (−14.1%) | 0.686 (−9.6%) | 0.646 (−9.0%) |
| LibFM | 0.837 (−10.9%) | 0.763 (−9.3%) | 0.629 (−14.5%) | 0.587 (−15.5%) | 0.673 (−8.6%) | 0.648 (−8.7%) |
| LibFM + TransE | 0.853 (−9.2%) | 0.784 (−6.8%) | 0.649 (−11.8%) | 0.613 (−11.8%) | 0.694 (−8.6%) | 0.652 (−8.1%) |
| PER | 0.683 (−27.3%) | 0.639 (−24.0%) | 0.593 (−19.4%) | 0.577 (−17.0%) | 0.637 (−16.1%) | 0.568 (−20.0%) |
| CKE | 0.761 (−18.9%) | 0.703 (−16.4%) | 0.622 (−15.5%) | 0.562 (−19.1%) | 0.664 (−12.5%) | 0.643 (−9.6%) |
| W&D | 0.861 (−8.3%) | 0.791 (−5.9%) | 0.684 (−7.1%) | 0.645 (−7.2%) | 0.716 (−5.7%) | 0.654 (−7.9%) |
| RippleNet | 0.896 (−4.6%) | 0.812 (−3.4%) | 0.697 (−5.3%) | 0.651 (−6.3%) | 0.724 (−4.6%) | 0.675 (−4.9%) |
| KGCN | 0.905 (−3.6%) | 0.823 (−2.1%) | 0.705 (−4.2%) | 0.684 (−1.6%) | 0.736 (−3.3%) | 0.687 (−3.2%) |
| KGAT | 0.911 (−2.9%) | 0.825 (−1.9%) | 0.703 (−4.4%) | 0.685 (−1.4%) | 0.743 (−2.1%) | 0.696 (−2.0%) |
| NCF-CNN | 0.858 (−8.6%) | 0.781 (−7.1%) | 0.650 (−11.7%) | 0.612 (−11.9%) | 0.695 (−8.4%) | 0.651 (−8.3%) |
| KGNCF-polling | 0.887 (−5.5%) | 0.804 (−4.4%) | 0.698 (−5.2%) | 0.656 (−5.6%) | 0.721 (−5.0%) | 0.681 (−4.08%) |
| KGNCF-RNN | 0.902 (−3.9%) | 0.815 (−3.1%) | 0.704 (−4.3%) | 0.685 (−1.4%) | 0.731 (−3.7%) | 0.687 (−3.2%) |
| KGNCF-RRN | **0.939** | **0.841** | **0.736** | **0.695** | **0.759** | **0.710** |

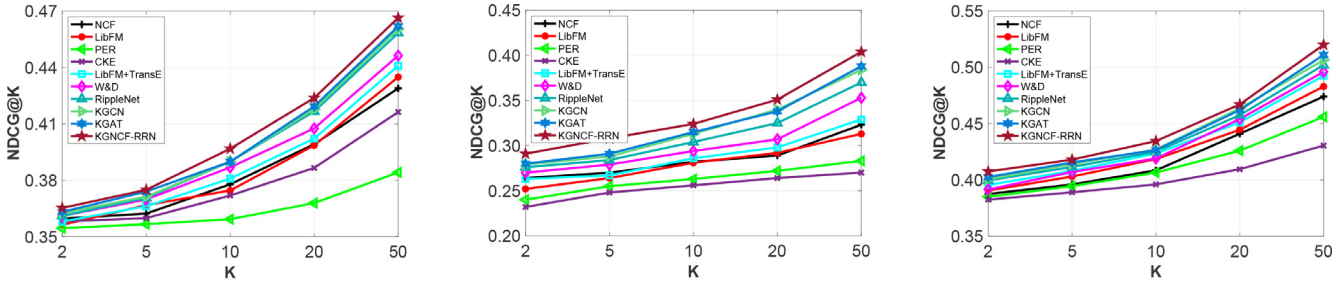


**Fig. 6.** The results of *NDCG@K* in top-K recommendation.

long-term dependencies of knowledge graph in an explicit way, so as to capture collaborative item relation signal effectively.

- CF-based methods that without using any KG information (*i.e.* LibFM and NCF) actually achieve better performance than the two KG-aware baselines CKE and PER in most cases. This result indicates that the hand-crafted meta-paths can not fully exploit the complex semantic interaction in knowledge graph, and user-defined meta-paths can also hardly be optimal in reality.
- LibFM + TransE outperforms LibFM, which demonstrates the benefit of the introduction of auxiliary KG for the recommendation in general. As a generic recommendation tool, Wide&Deep method achieve satisfactory performance, demonstrating that they can make well use of knowledge from KG into their algorithms.
- It is noteworthy that the recently proposed RippleNet model works well among these baselines. RippleNet adopts a preference propagation approach to exploit the potential preference in KG, which shows that capturing proximity information in the KG is essential for the recommendation. However, RippleNet is more sensitive to the density of datasets and can hardly capture the complex pairs interaction between users and items.
- It is noteworthy that the recently proposed RippleNet, KGCN and KAT work well among these baselines. Both RippleNet, KGCN and KAT are Structure-based methods that use breadth-first-search to obtain multi-hop neighbours of an entity in the KG. RippleNet adopts a preference propagation approach to exploit the potential preference in KG, which shows that capturing proximity information in the KG is essential for the recommendation. KGCN and KGAT are representative of inward aggregation that learns the representation of an entity by aggregating information from its multi-hop neighbours to mine users' potential preferences. All these methods calculate inner product or cosine similarity between user and item to obtain the matching score. However, such a simple way of matching has limitations in model expressiveness.
- We compare KGNCF-RRN with our proposed three simplifications. We can see from the Table 2 that removing knowledge graph degrades the model's performance as shown by NCF-CNN. Besides, the simple mean polling layer treat every paths with equal importance, cannot identify the saliency of each path (i.e., sequence) between an entity pair. Finally, the inferior performance of KGNCF-RNN justifies the effectiveness of the residual mechanism in long-term relational dependencies KG context modelling.

#### 4.4.2. Results in sparse scenarios

One major advantage of incorporating knowledge graph in KGNCF-RRN is to alleviate the sparsity problem of recommender systems. To study the effect of the knowledge graph embedding module in sparse scenarios, we vary the ratio of MovieLens-1M training set from $r = 20\%$ to $r = 100\%$ (while the validation and test set are kept fixed), and show the CTR prediction performance with *AUC*. The reason for choosing MovieLens-1M dataset is that MovieLens-1M have relatively sufficient training data than the other two, which make it easier in test to simulate the sparse scenario. We show the final results in Table 3. With the reducing of the training set, we observe that the performance of all methods deteriorate. When $r = 20\%$, the *AUC* score decreases by 17.3%, 12.2%, 16.1%, 16%, 11.7%, 15% and 11.9% for the baselines compared with the case when full training set is used ($r = 100\%$). In contrast, the *AUC* score of KGNCF-RRN only decreases by 7.5%, which shows that the proposed method can still maintain a decent performance even when the user-item interaction is sparse.

**Table 3**
Results of *AUC* on MovieLens-1M in CTR prediction with different ratios of training set *r*.

| Model | *r* | | | | |
|---|---|---|---|---|---|
| | 20% | 40% | 60% | 80% | 100% |
| LibFM | 0.692 | 0.748 | 0.783 | 0.813 | 0.837 |
| LibFM + TransE | 0.749 | 0.786 | 0.823 | 0.847 | 0.853 |
| NCF | 0.708 | 0.776 | 0.817 | 0.831 | 0.844 |
| PER | 0.574 | 0.657 | 0.664 | 0.671 | 0.683 |
| CKE | 0.672 | 0.721 | 0.739 | 0.741 | 0.761 |
| W&D | 0.732 | 0.803 | 0.84 | 0.851 | 0.861 |
| RippleNet | 0.789 | 0.792 | 0.826 | 0.843 | 0.896 |
| KGCN | 0.792 | 0.815 | 0.841 | 0.854 | 0.905 |
| KGAT | 0.787 | 0.809 | 0.847 | 0.867 | 0.911 |
| KGNCF-RRN | **0.869** | **0.882** | **0.894** | **0.923** | **0.939** |

### 4.5. Parameter sensitivity

In this section, we investigate how the recommendation performance varies with the hyper-parameters in (1) the impact of feature map number, (2) impact of tradeoff balance parameter $\alpha$. (3) impact of path number.

*1) Impact of Feature Map Number.* The number of feature maps in each CNN layer decides the dimension of final prediction layer, and will finally affect the representation ability of our KGNCF-RRN. Fig. 7 shows the performance of KGNCF-RRN with respect to different numbers of feature maps. We can see that although there are some slight differences in the convergence curve, all the curves increase steadily and finally achieve similar performance. This result reflects the strong expressiveness and generalisation of using CNN under the KGNCF-RRN framework since dramatically increasing the number of parameters of a neural network does not lead to overfitting.

*2) Impact of Tradeoff Parameter* There is a critical balance parameter $\alpha$ in the final loss function (9), which decides the weights between the KG path embedding loss and the prediction loss. The larger value of $\alpha$ will put more weight on the KG path embedding loss. We show the results with the varying parameter $\alpha$ in Fig. 8. For KGNCF-RRN, the performance shows an apparent increase at first, however, the performance then starts to decrease, when we continuously increase the balance over $\alpha = 0.15$ on all datasets. The reason is that, when $\alpha = 0$, KGNCF-RRN degenerates to the NCF-CNN without any knowledge graph information. When we increase $\alpha$ from 0, we actually strengthen the KG path embedding for context modelling. However, a too large value of $\alpha$ would drive the objective function bias to the KG context modelling. Given the experimental finding, we set $\alpha = 0.15$ for all three datasets.

*3) Impact of path number.* We vary the dimension of path numbers to 8, 32, 64, 128 and 256. Fig. 9 shows the accuracy of KGNCF-RRN over different settings of the path numbers. The AUC score shows an apparent increase at first; the reason for this is that more paths can encode more useful information. However, the performance then starts to slowly decrease, when we continuously increase the number of path. This is intuitive as a too-large path size requires more data to prevent an overfitting problem.

### 4.6. Effect of Self-attention

As stated in 3.2.1, we use a self attention structure in the path-level interaction layer. Since paths serve as important interaction context, the attention weights provide explicit evidence to understand why an interaction happens. To see this, we select two users in the Movielens dataset as an illustrative example. We checked the attention scores of features to see how multiple paths influence each other. The part of attention scores is visualised in Fig. 10. We investigate What role do two paths 1) UMDM (user-movie-director-movie), 2) UMAM (user-movie-actor-movie) play. We can see that each interaction corresponds to a unique attention distribution, summarizing the contributions of the meta-paths. The first interaction mainly relies on the meta-paths UMAM, while the second interaction relies more on the meta-path UMDM.

## 5. Related work

### 5.1. KG-based recommendation

We briefly introduce two main categories of knowledge-aware recommendation methods.

#### 5.1.1. Embedding-based methods

Prior efforts [52,46,38,39,29] leverage KG embedding as a regulariser to guide the learning of user and item latent vector, while direct user-item connections are used to optimise the objective function for recommendation. For example, CKE [52] and DKN [39] generate semantic embeddings from KG via knowledge graph embedding (KGE) techniques, and then incorporate them into recommenders – matrix factorization (MF) [25] and neural MF [13], respectively. Despite their great success, they only take direct connections between entities into consideration, while forgoing compositional relations within multi-hop paths. Such major limitation makes them a suboptimal solution to the recommendation task.

#### 5.1.2. Path-based methods

Another line of research introduces meta-paths [50,54,14] and paths [46,33] to directly infer user preferences. For instance, FMG [54] and MCRec [14] encode paths as embeddings to represent the user-item connectivity w.r.t meta-paths; RippleNet [38] recently construct ripple set (*i.e* high-order neighbouring items derived from KG) for each user to enrich her representations. While explicitly modelling high-order connectivity, it is highly challenging in real-world recommendation scenarios because most of these methods require extensive domain knowledge to define meta-paths or labor-intensive feature engineering to obtain qualified paths. Moreover, the scale of paths can easily reach millions or even larger when a large number of KG entities are involved, making it prohibitive to efficiently transfer knowledge.

### 5.2. Neural collaborative filtering

There are two types of methods for implementing an effective neural collaborative filtering model [13,6,12]. One is based on representation learning and the other one is based on matching function learning.

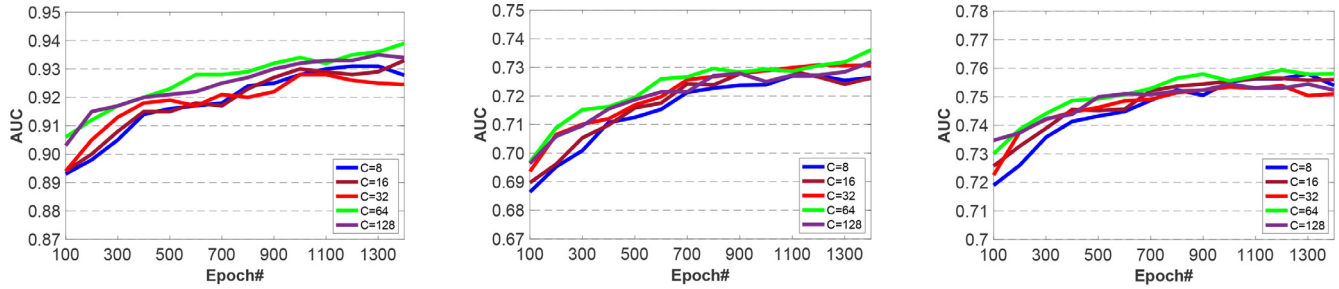


**Fig. 7.** Performance of KGNCF-RRN *w.r.t.* different numbers of feature maps per convolutional layer (denoted by $C$) in each epoch.

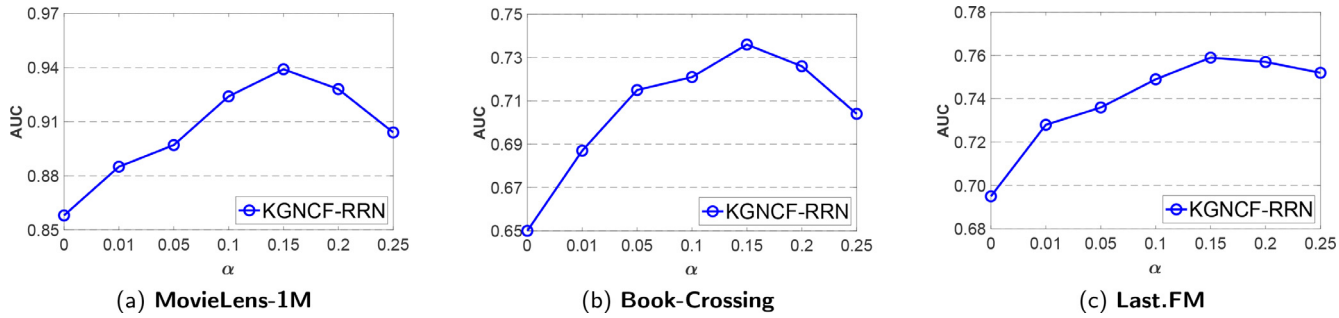


**Fig. 8.** Impact of balance parameter $\alpha$.

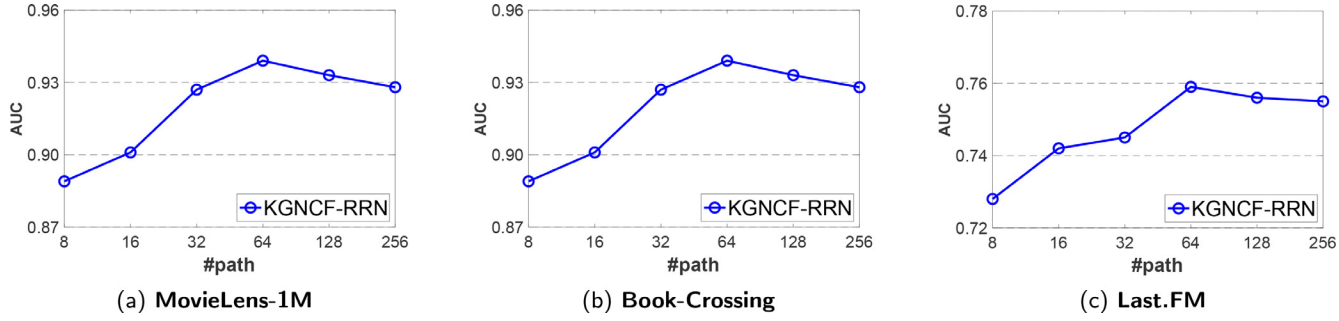


**Fig. 9.** Impact of the path numbers.

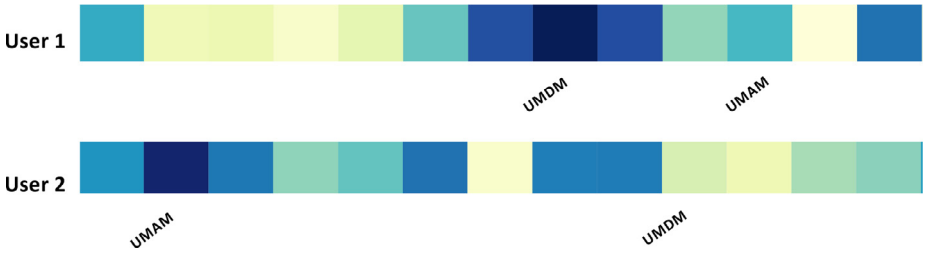


**Fig. 10.** Attention score visualization.

*5.2.1. Collaborative filtering based on representation learning*

Since early work Funk-SVD [7] win the Netflix Prize competition, matrix factorisation based methods have become the mainstream of recommendation research over the past ten years. [26,17,21,16]. These works try to assist matrix factorisation by incorporating auxiliary information, such as time, social relationship, text description, and location. Recently, there are also some researches proposed to use deep learning methods for collaborative filtering based on representation learning. AutoRec [30] is the first model that try to use autoencoder to learn user and item representation. DMF [48] propose matrix factorisation model with a two pathway neural network architecture to factorise rating matrix and learn user/item representations into low dimensional vectors. Overall, representation learning-based methods learn rep-

resentation in different ways and can flexibly incorporate with auxiliary data such as images, text descriptions, demographic information and so on. However, they still resort to the dot product or cosine similarity when predicting matching score.

*5.2.2. Collaborative filtering based on matching function learning*

Neural collaborative filtering (NCF) [13] is a recently proposed framework that unifies MF and MLP in one model, which replaces the dot product used in vanilla MF with a neural network to learn the matching function between users and items. NNCF [1] is a variant of NCF that takes user neighbors and item neighbors as inputs. To better capture pairwise correlations user and item dimension, ConvNCF [12] replace concatenation used in NeuMF to an outer product operation. Other than NCF, there are also many other works that introduce auxiliary information to learn the matching function. For example, Wide&Deep [5] learn the matching function by adapting LR and MLP with the help of categorical features of user and item. In this paper, we focus on neural collaborative filtering with auxiliary knowledge graph.

## 6. Conclusion

In this paper, we develop an end-to-end neural architecture KGNCF-RRN that jointly incorporates the knowledge graph structure and user-item interaction in a unified neural network model for recommendation. Specifically, we propose Residual Recur rent Network (RRN) to construct path embedding, which integrate recur rent neural networks (RNNs) with residual learning to efficiently bridge the gaps between entities and capture the long-term relational dependencies within KGs. Besides, a two channel neural interaction model is designed to encode both of the long-term m dependencies of KG in path embedding and complex matching interaction of user-item. All the parameters in KGNCF-RRN are optimised in a joint manner. Thus, the proposed model can capture both the enriched semantic information and the complex hidden relationships between users and items for recommendation. In the meantime, the high-order connectivity of KG is seamlessly incorporated into this recommendation framework. Finally, extensive experimental results on three real-world datasets clearly show the effectiveness of our proposed model.

## CRediT authorship contribution statement

**Lei Sang:** Conceptualization, Data curation, Software, Writing - original draft. **Min Xu:** Conceptualization, Methodology, Validation, Writing - review & editing. **Shengsheng Qian:** Methodology, Formal analysis, Supervision, Writing - review & editing. **Xindong Wu:** Resources, Project administration, Funding acquisition.

## Declaration of Competing Interest

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.

## Acknowledgments

This work was supported in part by the programs for Innovative Research Team in University of the Ministry of Education under Grant IRT_17R32, National Key Research and Development Program of China, under grant 2016YFB1000901, and in part by the National Natural Science Foundation of China under Grant 61673152 and Grant 91746209.

## References

[1] T. Bai, J.R. Wen, J. Zhang, W.X. Zhao, A neural collaborative filtering model with interaction-based neighborhood, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, 2017, pp. 1979–1982.
[2] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, in: Advances in Neural Information Processing Systems, 2013, pp. 2787–2795.
[3] Y. Cao, X. Wang, X. He, T.S. Chua, et al., Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences, 2019, arXiv preprint arXiv:1902.06236..
[4] S. Chen, Y. Peng, Matrix factorization for recommendation with explicit and implicit feedback, Knowledge-Based Systems 158 (2018) 109–117.
[5] H. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. , G. Anderson, G. Corrado, W. Chai, M. Ispir, R. Anil, Z. Haque, L. Hong, V. Jain, X. Liu, H. Shah, Wide & deep learning for recommender systems, in: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, DLRS@RecSys 2016, Boston, MA, USA, September 15, 2016, 2016, pp. 7–10..
[6] Z.H. Deng, L. Huang, C.D. Wang, J.H. Lai, P.S. Yu, Deepcf: A unified framework of representation learning and matching function learning in recommender system, 2019, arXiv preprint arXiv:1901.04704..
[7] S. Funk, Netflix update: Try this at home, 2006, https://sifter.org/simon/journal/20061211.html. accessed 12-June-2019..
[8] G. Guo, H. Qiu, Z. Tan, Y. Liu, J. Ma, X. Wang, Resolving data sparsity by multi-type auxiliary implicit feedback for recommender systems, Knowledge-Based Systems 138 (2017) 202–207.
[9] L. Guo, Z. Sun, W. Hu, Learning to exploit long-term relational dependencies in knowledge graphs, in: ICML, 2019, pp. 2505–2514.
[10] M. Gutmann, A. Hyvärinen, Noise-contrastive estimation: A new estimation principle for unnormalized statistical models, in: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, 2010, pp. 297–304.
[11] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[12] X. He, X. Du, X. Wang, F. Tian, J. Tang, T.S. Chua, Outer product-based neural collaborative filtering, 2018, arXiv preprint arXiv:1808.03912..
[13] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T.S. Chua, Neural collaborative filtering, in: Proceedingsofthe26thInternationalConferenceonWorldWideWeb,International WorldWideWebConferencesSteeringCommittee,2017,pp.173–182.
[14] B. Hu, C. Shi, W.X. Zhao, P.S. Yu, 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 & Data Mining, 2018, pp. 1531–1540.
[15] G. Hu, Y. Zhang, Q. Yang, Conet: Collaborative cross networks for cross-domain recommendation, in: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22–26, 2018, 2018, pp. 667–676..
[16] L. Hu, A. Sun, Y. Liu, Your neighbors affect your ratings: on geographical neighborhood influence to rating prediction, in: Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval, 2014, pp. 345–354.
[17] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer (2009) 30–37.
[18] J. Li, C. Chen, H. Chen, C. Tong, Towards context-aware social recommendation via individual trust, Knowledge-Based Systems 127 (2017) 58–66.
[19] Y. Lin, Z. Liu, M. Sun, Y. Liu, X. Zhu, Learning entity and relation embeddings for knowledge graph completion, in: Twenty-ninth AAAI Conference on Artificial Intelligence, 2015..
[20] H. Liu, Y. Wang, Q. Peng, F. Wu, L. Gan, L. Pan, P. Jiao, Hybrid neural recommendation with joint deep representation learning of ratings and reviews, Neurocomputing 374 (2020) 77–85.
[21] H. Ma, An experimental study on implicit social recommendation, in: Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2013, pp. 73–82.
[22] C. Musto, P. Lops, M. de Gemmis, G. Semeraro, Semantics-aware recommender systems exploiting linked open data and graph-based features, Knowledge-Based Systems 136 (2017) 1–14.
[23] W. Pan, A survey of transfer learning for collaborative recommendation with auxiliary data, Neurocomputing 177 (2016) 447–453.
[24] S. Rendle, Factorization machines with libfm, ACM Transactions on Intelligent Systems and Technology (TIST) 3 (2012) 57.
[25] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme, Bpr: Bayesian personalized ranking from implicit feedback, in: Proceedings of the twenty-fifth Conference on Uncertainty in Artificial Intelligence (AUAI), AUAI Press, 2009 pp. 452–461..
[26] R. Salakhutdinov, A. Mnih, Bayesian probabilistic matrix factorization using markov chain monte carlo, in: Proceedings of the 25th International Conference on Machine Learning, 2008, pp. 880–887.
[27] L. Sang, M. Xu, S. Qian, M. Martin, P. Li, X. Wu, Context-dependent propagating based video recommendation in multimodal heterogeneous information networks, IEEE Transactions on Multimedia (2020).
[28] L. Sang, M. Xu, S. Qian, X. Wu, Multi-modal multi-view bayesian semantic embedding for community question answering, Neurocomputing 334 (2019) 44–58.

[29] L. Sang, M. Xu, S. Qian, X. Wu, Knowledge graph enhanced neural collaborative recommendation, Expert Systems with Applications 113992 (2020).
[30] S. Sedhain, A.K. Menon, S. Sanner, L. Xie, Autorec: Autoencoders meet collaborative filtering, in: Proceedings of the 24th International Conference on World Wide Web, 2015, pp. 111–112.
[31] T. Shen, T. Zhou, G. Long, J. Jiang, S. Pan, C. Zhang, Disan: Directional self-attention network for rnn/cnn-free language understanding, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.
[32] Y. Sun, J. Han, X. Yan, P.S. Yu, T. Wu, Pathsim: Meta path-based top-k similarity search in heterogeneous information networks, Proceedings of the VLDB Endowment 4 (2011) 992–1003.
[33] Z. Sun, J. Yang, J. Zhang, A. Bozzon, L.K. Huang, C. Xu, Recurrent knowledge graph embedding for effective recommendation, in: Proceedings of the 12th ACM Conference on Recommender Systems, 2018, pp. 297–305.
[34] Y. Tay, L.A. Tuan, S.C. Hui, Latent relational metric learning via memory-based a ention for collaborative ranking, in: Proceedings of the 27th International Conference on World Wide Web, 2018.
[35] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, &Lstrok, Kaiser, I. Polosukhin, Attention is all you need, in: Advances in Neural Information Processing Systems(NIPS), 2017, pp. 5998–6008..
[36] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, 2017, arXiv preprint arXiv:1710.10903..
[37] C.D. Wang, Z.H. Deng, J.H. Lai, S.Y. Philip, Serendipitous recommendation in e-commerce using innovator-based collaborative filtering, IEEE Transactions on Cybernetics (2018) 1–15.
[38] H. Wang, F. Zhang, J. Wang, M. Zhao, W. Li, X. Xie, M. Guo, Ripple network: Propagating user preferences on the knowledge graph for recommender systems, CIKM (2018).
[39] H. Wang, F. Zhang, X. Xie, M. Guo, Dkn: Deep knowledge-aware network for news recommendation, in: Proceedings of the 2018 World Wide Web Conference on World Wide Web, International World Wide Web Conferences Steering Committee, 2018, pp. 1835–1844..
[40] H. Wang, F. Zhang, M. Zhao, W. Li, X. Xie, M. Guo, Multi-task feature learning for knowledge graph enhanced recommendation, in: The World Wide Web Conference, 2019, pp. 2000–2010.
[41] H. Wang, M. Zhao, X. Xie, W. Li, M. Guo, Knowledge graph convolutional networks for recommender systems, in: The World Wide Web Conference, 2019, pp. 3307–3313.
[42] Q. Wang, Z. Mao, B. Wang, L. Guo, Knowledge graph embedding: A survey of approaches and applications, IEEE Transactions on Knowledge and Data Engineering 29 (2017) 2724–2743.
[43] X. Wang, X. He, Y. Cao, M. Liu, T.S. Chua, Kgat: Knowledge graph attention network for recommendation, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 950–958.
[44] X. Wang, X. He, L. Nie, T.S. Chua, Item silk road: Recommending items from information domains to social users, in: Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval, 2017, pp. 185–194.
[45] X. Wang, D. Wang, C. Xu, X. He, Y. Cao, T. Chua, Explainable reasoning over knowledge graphs for recommendation, in: AAAI, 2019..
[46] X. Wang, D. Wang, C. Xu, X. He, Y. Cao, T.S. Chua, Explainable reasoning over knowledge graphs for recommendation, 2018, arXiv preprint arXiv:1811.04540..
[47] L. Wu, P. Sun, R. Hong, Y. Ge, M. Wang, Collaborative neural social recommendation, IEEE Transactions on Systems, Man, and Cybernetics: Systems (2018) 1–13.
[48] H.J. Xue, X.Y. Dai, J. Zhang, S. Huang, J. Chen, Deep matrix factorization models for recommender systems, in: Proceedings of the 26th International Joint Conference on Artificial Intelligence, 2017.
[49] Z. Yang, W. Chen, J. Huang, Enhancing recommendation on extremely sparse data with blocks-coupled non-negative matrix factorization, Neurocomputing 278 (2018) 126–133.
[50] X. Yu, X. Ren, Q. Gu, Y. Sun, J. Han, Collaborative filtering with entity similarity regularization in heterogeneous information networks, in: International Joint Conferences on Artificial Intelligence (IJCAI) 27, 2013.
[51] X. Yu, X. Ren, Y. Sun, Q. Gu, B. Sturt, U. Khandelwal, B. Norick, J. Han, Personalized entity recommendation: A heterogeneous information network approach, in: Proceedings of the 7th ACM International Conference on Web Search and Data Mining, 2014, pp. 283–292.
[52] F. Zhang, N.J. Yuan, D. Lian, X. Xie, W.Y. Ma, Collaborative knowledge base embedding for recommender systems, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 353–362.
[53] Y. Zhang, Q. Ai, X. Chen, W.B. Croft, Joint representation learning for top-n recommendation with heterogeneous information sources, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, 2017, pp. 1449–1458.
[54] H. Zhao, Q. Yao, J. Li, Y. Song, D.L. Lee, Meta-graph based recommendation fusion over heterogeneous information networks, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017, pp. 635–644.

**Lei Sang** is currently pursuing the Ph.D. degree with the School of Computer Science and Information Engineering, Hefei University of Technology, China, and also with the Faculty of Engineering and Information Technology, University of Technology, Sydney, Ultimo, NSW, Australia. His current research interests include natural language processing and recommender system.

**Min Xu** received the B.E. degree from University of Science and Technology of China, in 2000, M.S degree from National University of Singapore in 2004 and Ph.D. degree from University of Newcastle, Australia in 2010. She is currently a Senior Lecturer at University of Technology, Sydney. Her research interests include multimedia data analytics, pattern recognition and computer vision. She has published over 100 research papers in high quality international journals and conferences.

**Shengsheng Qian** received the B.E. degree from the Jilin University, Changchun, China, in 2012, and the Ph.D. degree in pattern recognition and intelligent systems from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2017. He is currently an Assistant Professor with the Institute of Automation, Chinese Academy of Sciences. His current research interests include social media data mining and social event content analysis.

**Xindong Wu** received the Ph.D. degree in artificial intelligence from The University of Edinburgh, Edinburgh, U.K. He is a professor of computer science at the University of Louisiana at Lafayette, USA. His current research interests include data mining, knowledge-based systems, and Web information exploration. He is the Steering Committee chair of IEEE International Conference on Data Mining (ICDM). He is the editor-in-chief of Knowledge and Information Systems (KAIS) and ACM Transactions on Knowledge Discovery from Data (TKDD). He is a fellow of IEEE and the AAAS