Expert Systems With Applications 164 (2021) 113992

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Knowledge graph enhanced neural collaborative recommendation

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

[a] Key Laboratory of Knowledge Engineering with Big Data (Hefei University of Technology), Ministry of Education, Hefei 230009, 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] Mininglamp Academy of Sciences, Mininglamp Technology, Beijing, 100084, China

ARTICLE INFO

Keywords:
Recommendation system
Knowledge graph
Neural collaborative filtering
Graph convolutional networks
Attention mechanism

ABSTRACT

Existing neural collaborative filtering (NCF) recommendation methods suffer from severe sparsity problem. Knowledge Graph (KG), which commonly consists of fruitful connected facts about items, presents an unprecedented opportunity to alleviate the sparsity problem. However, pure NCF models can hardly model the high-order connectivity in KG, and ignores complex pairwise correlations between user/item embedding dimensions.

To address these problems, we propose a novel Knowledge graph enhanced Neural Collaborative Recommendation (K-NCR) framework, which effectively combines user–item interaction information and auxiliary knowledge information for recommendation task into three parts: (1) For items, the proposed propagating model learns the representation of item entity. It recursively aggregates information from its multi-hop neighbours in KG, and employs an attention mechanism to discriminate the importance of the relation type to mine users' potential preferences. (2) For users, another heterogeneous attention weights are leveraged to strengthen the embedding learning of users. (3) The user and item embeddings are then fed into a newly designed two-dimensional interaction map with convolutional hidden layers to model the complex pairwise correlations between their embedding dimensions explicitly. Extensive experimental results on three benchmark datasets demonstrate the effectiveness of our K-NCR framework.

1. Introduction

As the amount of data from different platforms grows at an unprecedented rate, the recommender system is becoming more and more crucial to help users discover personalised content of interest from these ever-growing corpus of data. For example, there are 11,895 movies released around the world in 2018.[1] It is impossible for users to watch all available movies to identify their interested ones. Due to their extraordinary capability of exploiting collective wisdom and experiences, Collaborative Filtering (CF) algorithms, especially Matrix Factorisation (MF) algorithms — a technique that predicts users' personalised preference from user–item interactions only — has proven to be the most widely used recommendation technology (Wang, Deng et al., 2018). Matrix factorisation assumes that there are some latent factors that exist behind the interactions between users and items. Though widely studied in the past, traditional matrix factorisation suffers from the severe sparsity problem of user–item interaction (Sun et al., 2019).

Inspired by the recent popularity of deep learning, Neural Collaborative Filtering (NCF) is proposed, as a new class of CF methods, cast the traditional MF algorithm into an overall neural framework (Deng et al., 2019; He, Du et al., 2018; He et al., 2017). Generally speaking, there are two key components in learnable NCF models: (1) embedding, which transforms users and items to vectorised representations, and (2) interaction modelling, which reconstructs historical interactions based on the user/item embeddings. For example, in He et al. (2017), the authors proposed to replace the MF interaction function of inner product with nonlinear deep neural networks. The translation-based CF models were proposed to use Euclidean distance metric as the interaction function (Tay et al., 2018). However, these methods may not be sufficient to yield satisfactory embeddings for CF. The possible reason is that most existing methods build the embedding function with the descriptive features only (e.g., ID or attributes). As a result, when the interaction function goes deeper to capture complex user–item relationship, these methods tend to overfit and make the sparsity problem even worse.

* Corresponding author.
E-mail addresses: lei.sang@student.uts.edu.au (L. Sang), Min.Xu@uts.edu.au (M. Xu), shengsheng.qian@nlpr.ia.ac.cn (S. Qian), xwu@hfut.edu.cn (X. Wu).

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

https://doi.org/10.1016/j.eswa.2020.113992
Received 17 November 2019; Received in revised form 6 September 2020; Accepted 9 September 2020
Available online 12 September 2020

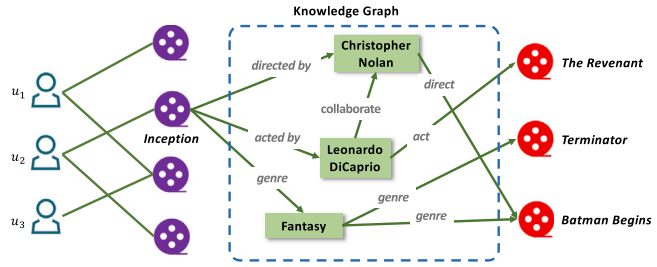


Fig. 1. Illustration of KG-enhanced movie recommender systems. The knowledge graph can provides extra user–item connectivity information, which are useful for alleviating the sparsity problem of target recommendation task.

Therefore, researchers proposed using auxiliary data in recommender systems to enrich the semantics of the user or item representation, such as social networks, attributes, and multimedia (Hu, Zhang et al., 2018; Luo et al., 2019; Qian et al., 2018; Symeonidis & Malakoudis, 2019; Zheng et al., 2018). Recently, Knowledge Graphs (KGs) have attracted increasing attention, which usually consist of fruitful connected facts about items (Cao et al., 2019; Sourabh & Chowdary, 2019; Wang, Zhang, Zhao et al., 2019). KG introduces semantic relatedness among items, which is a useful tool to help mine their latent connections and improve the quality of recommendation. Extra connectivity information derived from KG endows recommender systems the ability of reasoning and alleviates sparsity problem (Wang, He et al., 2019). Taking movie recommendation as an example in Fig. 1, a user 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.

The prior works on KG-based recommendation can be roughly categorised into two categories considering the path and embedding fashion. (1) Path-based methods use predefined meta paths with specific connectivity patterns to capture different semantic similarities between users and items carried in KGs. Path-based methods exploit KG in an intuitive way, but they fail to automatically uncover and reason on unseen connectivity patterns, since they rely heavily on hand-crafted meta-paths. (2) Another widely researched line is embedding-based methods, which pre-process a KG with knowledge graph embedding (KGE) (Wang, Mao et al., 2017) algorithms, such as TransE (Bordes et al., 2013) and TransR (Lin et al., 2015), and then leverage the learned entity embeddings to regularise the representations of items. The high-order connectivity denotes the path that reaches users from any entity node with the path length larger than 1 as the above movie example shows. However, these methods only consider direct relations between entities, rather than the high-order connectivity paths.

Recent developments of graph neural networks (Hamilton et al., 2017; Kipf & Welling, 2017; Ying et al., 2018) try to automatically capture high-order structure information in a graph, which has the potential of achieving the goal but has not been explored much for KG-based recommendation. Another key deficiency is that they model each interaction in KG as an independent instance and do not consider their relations, which makes them insufficient to discriminate the various user preference revealed by different relation type in KG. As shown in Fig. 1, movie *"Inception"* connects to movie *"Batman Begins"* with two kinds of relations, which can be caused by user's different interest propensity (prefer a movie with the same "genre" or "director"). Moreover, these methods are proposed for a single recommendation task (He et al., 2017) with only KG data. And they fail to study knowledge enhanced recommendation using deep neural networks in an end-to-end way.

To this end, we investigate how to utilise neural collaborative based model for both user–item and auxiliary knowledge graph data, and enable the knowledge to enrich the CF model with sparsity user–item interaction. Then three key challenges arise in this investigation. (1) how to learn item representation from complex heterogeneous structured KG. Traditional KGE regularisation only considers direct relations between entities, rather than the multi-hop relation paths. (2) how to model user profile from weighted items. Existing user profile modelling methods can be limited by its assumption that all historical items of a user profile contribute equally in estimating the similarity between the user profile and a target item. Intuitively, a user interacts with multiple items in the past, but it may not be true that these interacted items reflect the user's interest to the same degree. (3) how to capture the complex interactions between users and items, which is not exploited by the prior shallow interaction-based methods. The traditional inner product matching function assumes that the dimensions of users and items embedding are independent with each other, and contribute equally for the prediction of all data points (He et al., 2017). To some extent, this assumption is impractical, since the embedding dimensions could be interpreted as certain properties of items (He, Du et al., 2018; Zhang et al., 2014), which are not necessarily to be independent.

In this paper, we propose a novel Knowledge Graph enhanced Neural Collaborative Recommendation(K-NCR) approach. K-NCR tackles above three challenges in a recommendation scenario as shown in Fig. 3. The proposed K-NCR has three key modules: (1) For item modelling, we exploit the high-order structural proximity among entities in KG instead of the one-hop local neighbour structure. In particular, we capture the influence diffusion process among items with multi-hop based on Graph Convolutional Network (GCN), which is capable of stimulating the propagation of user preferences over the set of knowledge entities by iteratively and automatically extending potential interests along with links in the knowledge graph. Moreover, we employ an attention mechanism to discriminate the importance of the relation type to mine users' potential preferences. (2) For user modelling, to alleviate the limitation of traditional mean-based aggregator in NCF, the varying importance weights are learned from the interacted items with an attention network. These weights can distinguish historical item in a user profile is playing a more important role for the user preference modelling. (3) To capture the complex interaction between user and item embeddings, our proposal of using outer product above the user and item embedding layer results in 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. Given the proposed neural architecture, we design a joint learning framework that allows the KG embedding part and the neural collaborative recommendation part to enhance each other mutually.

The main contributions of this paper are as follows:

1. We are the first to combine KG structure information with collaborative recommendation in an end-to-end neural style.
   We highlight the importance of explicitly modelling (1) the high-order connectivity of knowledge graph to provide a better recommendation with item side information and (2) the pairwise correlations between the dimensions of the embedding space in the matching function.
2. We propose a new recommendation framework K-NCR, which encodes the high-order connectivities of KG and complex matching interaction in an explicit and end-to-end manner by performing embedding propagation and outer product-based convolutional NCF, respectively.
3. We conduct empirical studies on three million-size real-world dataset.
   Extensive experiment results demonstrate the state-of-the-art performance of K-NCR and its effectiveness in improving the embedding and matching quality for the final prediction (see Table 1).

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)
$\mathcal{E} = \{e_1, e_2, \ldots\}$	Set of entities
$\mathcal{R} = \{r_1, r_2, \ldots\}$	Set of relations
$\mathcal{N}_i$	Neighbours of entity e_i
$\mathcal{N}_i^r$	Neighbours of entity e_i under relation r
$\mathbf{h}_i^{(l)}$	Entity embedding in layer l
$\mathbf{k}_u \in \mathbb{R}^S$	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

2. Preliminary

Before introducing the proposed approach, we first define the task of knowledge graph enhanced recommendation, and then shortly recapitulate the standard neural collaborative filtering, highlighting its limitations for dealing with the task.

2.1. Problem formulation

In the study of KG-enhanced recommendation, we have a target recommendation domain with user–item interaction data and an auxiliary knowledge graph data domain . For the data in the target recommendation domain, we have a set of users $\mathcal{U} = \{u_1, u_2, \ldots, u_M\}$ and a set of items $\mathcal{I} = \{i_1, i_2, \ldots, i_N\}$, where M and N are the number of users and items, respectively. The user–item interaction data is usually presented as a bipartite graph. In particular, the matrix of user–item interaction $Y \in \mathbb{R}^{M \times N}$ is constructed according to users' implicit feedback as follows:

$$y_{ui} = \begin{cases} 1 & \text{if interaction } (u, i) \text{ is observed;} \\ 0 & \text{otherwise.} \end{cases}$$

Here an observed interaction ($y_{ui} = 1$) implies that user u engaged with item i, e.g., users browsed a 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, it could be that user u is not aware of the item since the overwhelming amount of items available in a system. The unobserved entries of user–item interaction matrix can be just missing data, and there is a natural scarcity of negative feedback, which poses challenges in learning from these implicit data (Xiong et al., 2018). The problem in target recommendation domain can be formulated as evaluating the scores of unobserved entries in Y.

In the auxiliary data domain, we have access to a knowledge graph $\mathcal{G}$, which is a directed graph composed of subject-relation-object triple facts (h, r, t), which indicates a fact that there is a relationship r from head entity h to tail entity t. For example, the triple (*Christopher Nolan*, *film.director.film*, *Batman Begins*) states the fact that *Christopher Nolan* directs the film *Batman Begins*. KG provides deep factual knowledge and rich semantics on items. More important, by exploring the interlinks within a KG, the connectivity between users and items reflects their underlying relationships, which are complementary to the sparsity user–item interaction (Wang, He et al., 2019).

Given the user–item interaction matrix Y as well as the knowledge graph $\mathcal{G}$, our goal is to predict the potential user interests for the unobserved feedbacks. Specifically, we try to learn a prediction function $\hat{y}_{ui} = \mathcal{F}(u, i|\theta, Y, \mathcal{G})$ to calculate the probability that user u may choose item i, where θ is the model parameters of function $\mathcal{F}$. Notations used throughout the article can be found in Table Table 1.

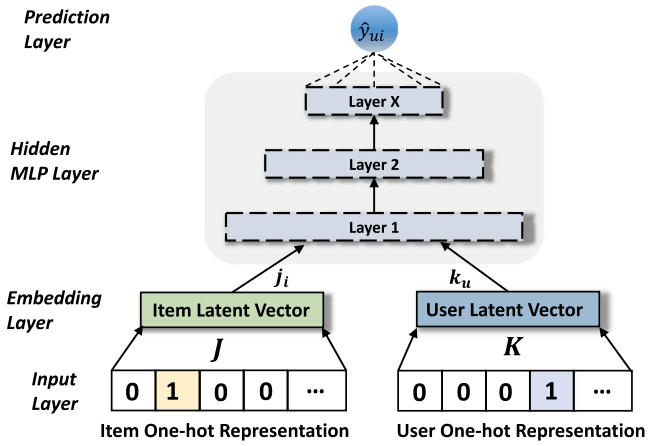


Fig. 2. Neural Collaborative Filtering (NCF): MF as a shallow neural network model.

2.2. Neural collaborative filtering framework

Collaborative filtering (CF) is the fundamental methods for personalised recommendation systems, of which model based Matrix Factorisation (MF) is one of the simplest yet effective implementations (Koren et al., 2009). 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:

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

where $\mathbf{k}_u \in \mathbb{R}^S$ and $\mathbf{j}_i \in \mathbb{R}^S$ denote the latent vector for user u and item i, respectively. And S denotes the dimension of the latent space, which is much smaller than the size of users or items. 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 be deemed as a linear model of latent factors (Wang, He et al., 2017).

To overcome the limitation of linear MF model, it is natural to model the interaction between users and items with a two-way neural network. Neural Collaborative Filtering (NCF) (He et al., 2017) is the representative works as shown in Fig. 2. Let M_u and M_i denote the feature information, or just one-hot representation of user u and item i. The prediction function is defined as:

$$\hat{y}_{ui} = f(K^T M_u, J^T M_i | \mathcal{U}, \mathcal{I}, \theta) \tag{2}$$

where $K \in \mathbb{R}^{M \times S}$ and $J \in \mathbb{R}^{N \times S}$ are the embedding matrix for user features and item features, respectively; function $f(\cdot)$ represents the multilayer perceptron, and θ is the parameters of this network. Traditional MF-based recommendation method can be viewed as a simple case of NCF, while the non-linear MLP hidden layer of NCF can model more complex user–item interaction.

Although using MLP to learn the matching function directly endows the model with great flexibility, there is no practical guarantee that the dimension correlations can be effectively captured with current optimisation techniques, especially with complex user and item interaction (He, Du et al., 2018). The case can be even worse if we take the auxiliary attributes information (such as Knowledge Graph, and multimedia content) into account. To overcome shortages of the above issue and further improve the performance of CF methods, we incorporate them under the proposed recommendation K-NCR framework.

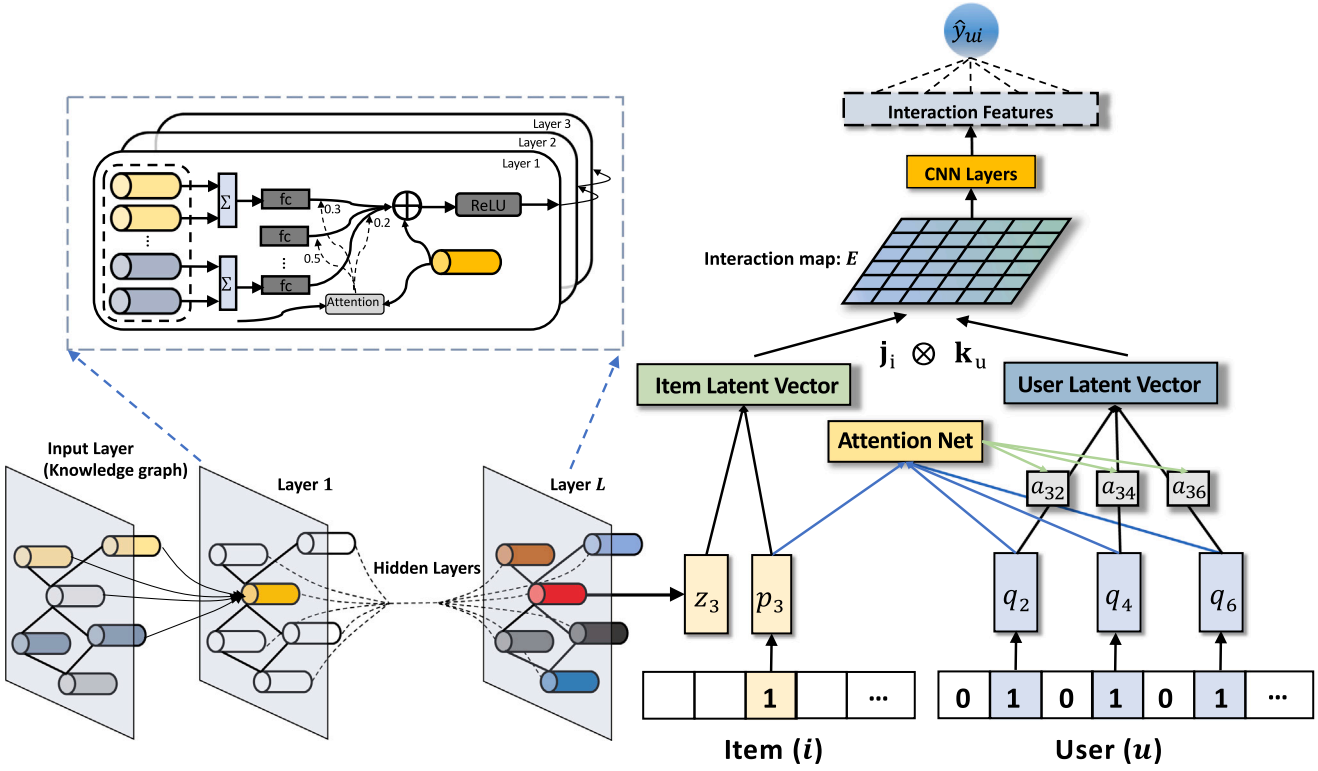


Fig. 3. Overall architecture of K-NCR. There are three components in the framework: (1) an knowledge aggregating-based Item modelling (Left) (2) attention-based User Modelling (Low right) and (3) Convolutional NCF for item/user interaction modelling (Upper right).

3. Knowledge enhanced neural collaborative model

3.1. Framework

The goal of KG-enhanced recommendation is to select relevant information from the knowledge graph to assist the target recommendation prediction. In this paper, we propose a knowledge graph enhanced Neural Collaborative Recommendation (K-NCR), an end-to-end framework that utilises KG to alleviate the sparsity problem of recommender systems. We now present the proposed K-NCR model, and the architecture of which is illustrated in Fig. 3. There are three components in the framework: (1) knowledge propagation based item modelling, (2) attention based user preference modelling and (3) convolutional NCF for item/user interaction modelling.

3.2. Knowledge propagation based item modelling

Item modelling try to mine latent semantic embedding of items for recommendation. In the KG-enhanced recommendation, we assume each item's embedding is composed of two parts: a KG embedding part $\mathbf{z}$ that could be captured by the correlation phenomenon of users' interests in the Knowledge Graph. Besides, each user has his/her unique interest, which could not be modelled in the KG.

For entity embedding, a novel knowledge propagation model is proposed to embed an item entity in KG into a low-dimensional semantic embedding space, such that entities with neighbouring structure have similar vector representations. We propose a GCN (Graph Convolutional Networks) based entity embedding method, which is a prominent technique for feature representation learning in various graph structured data. Traditional KGE methods only consider the local structure of an entity, which is far from modelling users' varying interest in the knowledge. Different from traditional KGE, we show how to use GCN explore entities' potential high-order structural proximity on KG entities with deep learning-based models. For every entity node in the KG, GCN encodes relevant information about its neighbourhood as a real-valued feature vector in an unsupervised manner. Noting that, each item $i \in \mathcal{I}$ have its corresponding entity e_i in the knowledge graph.

As shown in the left part of Fig. 3, one layer GCN encodes only information about immediate neighbours and L layers are needed to encode L-order neighbourhoods (*i.e.,* information about nodes at most L hops aways). Specially, representation propagation process of entity embedding captures high-order structural proximity among entities in a knowledge graph, which consists of two basic operation: (1) **Propagation**: We iteratively propagate an entity's contexts along links in a graph-structured KG and explore long-range dependencies among the surrounding entity nodes; (2) **Aggregation**: We then collectively aggregate feature of long-range neighbour entities to get the embedding of target entity. The key idea is to learn the iterative convolutional operation in graphs as a message-passing framework, where each convolutional operation means generating the current node representations from the aggregation of local neighbours in the previous layer:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}_i} g(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)})\right), \tag{3}$$

where $\mathbf{h}_i^{(l)} \in \mathbb{R}^{d^{(l)}}$ is the latent embedding of entity e_i in the lth layer of the neural network, and $\mathcal{N}_i$ is the neighbour entity of e_i. $g(\cdot,\cdot)$ is typically chosen to be a (message-specific) neural network-like function (Schlichtkrull et al., 2018).

Considering the heterogeneous structure of knowledge graph, entities are connected by different type of relations, so it is unreasonable to ignore this crucial node type information. To this end, we define the following simple knowledge aggregating process by considering various relation type:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\sum_{r \in R} \sum_{j \in \mathcal{N}_i^r} \frac{1}{c_{i,r}} W_r^{(l)} \mathbf{h}_j^{(l)} + W_0^{(l)} \mathbf{h}_i^{(l)} + b\right), \tag{4}$$

where W_r^l and b are the weight matrix and bias, respectively, σ is an activation function, e.g. a ReLU. $\mathcal{N}_i^r$ is the neighbour entity set of target

Algorithm 1 Propagating layer of KGE

Input: Knowledge graph $\mathcal{G} = (\mathcal{E}, \mathcal{R})$; depth L; weight matrices $W^l, \forall l \in \{1, \ldots, L\}$; non-linearity σ;
Output: Entity embedding $\mathbf{z}_i$;

```
for l = 1 ... L do
    for e_i ∈ E do
        t_r = AVG(h_j^(l)), j ∈ N_i^r;
        β_r ← softmax(ReLU(t_r ⊕ h_i^(l)));
        h_i^(l+1) ← σ(SUM(W^l β_r t_r) + h_i^(l));
    end for
end for
return z_i ← h_i^(L), ∀e_i ∈ E
```

entity i under relation $r \in \mathcal{R}$. $c_{i,r}$ denotes a task-specific normalising constant, and here we set it as $c_{i,r} = |\mathcal{N}_i^r|$, the number of neighbours with relation type r. Intuitively, Eq. (4) aggregates transformed hidden vectors of neighbouring entity with a normalised sum. Different from classical GCN, we propose a relation-specific transformations, which depends on the type of a relation edge. Besides, a self-connection relation edge is added to ensure that the initial representation $\mathbf{h}^{(l)}$ at layer l directly informs its new representation $\mathbf{h}^{(l+1)}$ at layer $l+1$.

3.2.1. Attention based aggregation

Eq. (4) uses a sum operator as the aggregation function, which takes the element-wise mean of the neighbour vectors. It assumes that all type of neighbour contribute equally to the representation of the target entity e_i. However, as mentioned before, strong and weak ties are mixed together in a KG, and users are likely to share more similar tastes with strong ties than weak ties. For example, a user may have more potential interests in the movies that share the same "star" with his or her historically liked ones, while another user may be more concerned about "genre" of movies. Thus, we perform an attention mechanism with a two-layer neural network to extract these neighbour entity node that is important to target entity e_i, and model their tie strengths, by relation attention β_r with $\mathbf{t}_r$ (sum of neighbour embedding with relation type r) and the target entity embedding $\mathbf{h}_i$, as below,

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\sum_{r\in\mathcal{R}} \beta_r \mathbf{t}_r + \mathbf{W}_0^{(l)} \mathbf{h}_i^{(l)} + b_0\right) \tag{5}$$

$$\mathbf{t}_r = \sum_{j\in\mathcal{N}_i^r} \frac{1}{c_{i,r}} \mathbf{W}_1^{(l)} \mathbf{h}_j^{(l)} \tag{6}$$

$$\beta_r^* = \mathbf{w}_2^T \cdot \sigma(\mathbf{W}_1 \cdot [\mathbf{t}_r \oplus \mathbf{h}_i^{(l)}] + \mathbf{b}_1) + b_2 \tag{7}$$

$$\beta_r = \frac{exp(\beta_r^*)}{\sum_{r\in N(r)} exp(\beta_r^*)} \tag{8}$$

where the β_r^* can be seen as the weight for relation type r, and $N(r)$ is the number of entity relation types.

Starting from a bottom layer of node representations as their node features, GCN stacks multiple convolutional operations to simulate the message passing of graphs. We get the final entity embedding $\mathbf{z}_i = \mathbf{h}_i^{(L)}$ for entity e_i after L layers aggregation. Therefore, both the information propagation process with graph structure and node attributes are well leveraged in GCNs. Typically, the embedding fed into the first layer $\mathbf{h}_i^{(0)}$ is chosen as a unique one-hot representation if no other features are present. Here, we further feed this one-hot representation to a single linear transformation and get a dense representation as the first layer entity node feature.

3.2.2. Unsupervised training methods

To learn predictive entity representations, we apply an unsupervised graph-based loss function to the output entity representation in a fully unsupervised way, and tune the parameters via stochastic gradient descent. Specifically, we feed both positive and negative pairs to the loss model, and try to push positive nearby entity to the target entity while pushing negative disparate entity away from it in the low-dimensional vector space:

$$J_{\mathcal{G}}(\mathbf{z}_u) = -\log\left(\sigma(\mathbf{z}_u^\top \mathbf{z}_v)\right) - Q \cdot \mathbb{E}_{v_n\sim P_n(v)} \log\left(\sigma(-\mathbf{z}_u^\top \mathbf{z}_{v_n})\right), \tag{9}$$

where v denotes the paired positive entity that co-occurs in the context of target entity u within fixed-length random walk, and σ is the sigmoid function. In order to sample the negative context entity v_n, we also define sampling distribution P_n based on the node popularity, and Q denotes the number of negative samples.

3.2.3. Item latent vector

In KG-enhanced recommendation, we assume each item's latent vector representation is consists of two parts: an entity embedding part $\mathbf{Z}$ that could be captured by fruitful facts and connections about items in the knowledge graph. Besides, each item has its unique character, and we use the embedding matrix $\mathbf{P}$ to denote items' individual latent property. The learned aggregated entity embedding $\mathbf{z}_i \in \mathbf{Z}$ could complement the item embedding $\mathbf{p}_i \in \mathbf{P}$, thus organically alleviating sparsity problem by exploiting the extended information of knowledge graph. Then, each item i's latent vector $\mathbf{j}_i$ can be denoted as the sum of these two parts:

$$\mathbf{j}_i = \mathbf{z}_i + \mathbf{p}_i \tag{10}$$

3.3. Attention-based user modelling

User modelling aims to learn user latent factors, denoted as $\mathbf{k}_u \in \mathbb{R}^d$ for user u, which characterises a user's profile with his/her historically interacted items. After that, we can recommend new items that are highly related to the user's profile. In this section, an item aggregation model is proposed to learn factors from user–item interaction graphs, as shown in the right part in Fig. 3. To mathematically represent this aggregation, we use the following function as:

$$\mathbf{k}_u = \sigma(\mathbf{W} \cdot Aggre_{items}(\{\mathbf{q}_j, \forall j \in C(u)\}) + \mathbf{b}) \tag{11}$$

where $C(u)$ denotes the set of items that user u has interacted with (or user u's neighbours in the user–item graph), $\mathbf{q}_j$ denotes the embedding vector for user interacted item j, and $Aggre_{items}$ is the items aggregation function. In addition, σ denotes non-linear activation function (*i.e.,* a rectified linear unit), and $\mathbf{b}$ and $\mathbf{W}$ are the bias and parameters of a neural network, respectively. It is worth noting that, we encode each item with two embedding vectors $\mathbf{p}$ and $\mathbf{q}$ to differentiate its role of a prediction target or a historical interaction, which can also increase model expressiveness (He, He et al., 2018). Next we will discuss how to define the aggregation function $Aggre_{items}$.

Mean operator is one typical aggregation function for $Aggre_{items}$, where we take the element-wise mean of the vectors in $\{\mathbf{q}_j, \forall j \in C(u)\}$. The mean-based aggregator is a linear approximation of a localised spectral convolution (Kipf & Welling, 2017), as the following function:

$$\mathbf{k}_u = \sigma(\mathbf{W} \cdot \left\{\sum_{j\in C(u)} \frac{1}{|C(u)|} \mathbf{q}_j\right\} + \mathbf{b}) \tag{12}$$

However, the equal treatments on all historical items can limit the representation ability of user profile due to the fact that the influence of interactions on users may vary dramatically. Hence, we should allow interactions to contribute differently to a user's latent factor.

To alleviate the limitation of mean-based aggregator, inspired by attention mechanisms (He, He et al., 2018; Peng et al., 2019), an intuitive method is to tweak a trainable parameter that assigns a unique weight to the target item $\mathbf{p}_i$:

$$\mathbf{k}_u = \sigma(\mathbf{W} \cdot \left\{\sum_{j\in C(u)} a_{ij} \mathbf{q}_j\right\} + \mathbf{b}) \tag{13}$$

$$a_{ij} = f(\mathbf{p}_i, \mathbf{q}_j) \tag{14}$$

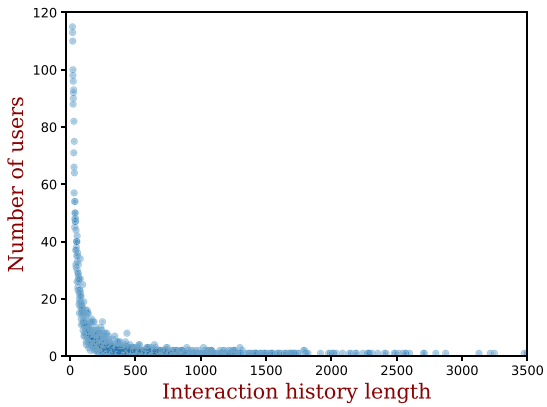


Fig. 4. The long-tail distribution of Movielens-20M dataset.

where a_{ij} denotes the attention weight of the interaction with $\mathbf{q}_j, j \in C(u)$ in contributing to user u's item-space latent factor when characterising user u's preference from the interaction history $C(u)$. We consider the item attention a_{ij} as a function with $\mathbf{p}_i$ and $\mathbf{q}_j$. The rationale is that the embedding vectors are supposed to encode the information of items, thus they can together to vote the weight of an interaction (i, j). Specifically, we parameterise the item attention a_{ij} with a two-layer neural network, which we call as the attention network:

$$a^*_{ij} = \mathbf{c}^T \cdot \sigma(\mathbf{W}_1 \cdot [\mathbf{q}_j \odot \mathbf{p}_i] + \mathbf{b}_1) + b_2 \tag{15}$$

where $\mathbf{W}_1$ is the corresponding input transformation's weight matrix that transform the input pair correlation into a hidden layer, and vector $\mathbf{c}^T$ aims to transform the hidden layer into the output attention weight. Larger value of attention weight means stronger influence to the user preference representation. We use the LeakyReLU nonlinearity (with negative input slope 0.2) as the activation function σ.

To make weight easily comparable across different items, we normalise the above attentive scores across all choice of j with softmax function:

$$\mathbf{k}_u = \sigma(\mathbf{W} \cdot \left\{ \sum_{j \in C(u)} a_{ij} \mathbf{q}_j \right\} + \mathbf{b}) \tag{16}$$

$$a_{ij} = \frac{exp(a^*_{ij})}{\sum_{j \in C(u)} exp(a^*_{ij})} \tag{17}$$

Then, the normalised attention is used to compute the weighted combination of the item features, to serve as the final output features for each user.

However, this kinds of attention still may encounter performance problem in practice due to the long-tail distribution of user interaction history length (*i.e.,* number of historical items users have interacted). The history length of users can vary in a large ranges as shown in Fig. 4. Attention mechanisms are commonly used in CV and NLP tasks, where the size of attention units does not vary much, such as regions in an image (Mun et al., 2017; Xu et al., 2015) and words in a sentence (Luong et al., 2015; Shen et al., 2018). Thus, attention weights can be properly normalised by softmax function and successively can reasonably explained based on the probability. However, when applying in the varying length of recommendation scenarios, the $L1$ normalisation of softmax function may overly punish the attention weights of active users with a long history (He, He et al., 2018). To address the problem, we leverage exponential smoothing to ease the weights punishment to active users, which can also achieve a variance reduction across all attention weights. Finally, the attention model can be modified to:

$$\mathbf{k}_u = \sigma(\mathbf{W} \cdot \left\{ \sum_{j \in C(u)} \alpha_{ij} \mathbf{q}_j \right\} + \mathbf{b}) \tag{18}$$

$$a_{ij} = \frac{exp(a^*_{ij})}{\sum_{j \in C(u)} exp(a^*_{ij})^{\beta}} \tag{19}$$

$$a^*_{ij} = \mathbf{c}^T \cdot \sigma(\mathbf{W}_1 \cdot [\mathbf{q}_j \odot \mathbf{p}_i] + \mathbf{b}_1) + b_2 \tag{20}$$

Here, β represents the smoothing exponent, which is a hyperparameter between 0 and 1. When β is smaller than 1, the value of denominator will be suppressed, as a result the attention weights will not be overly punished for active users. It was apparent that, when β is set to 1, it will degenerate to the original softmax function.

3.4. Convolutional NCF

After obtaining the latent vector representation of item $\mathbf{j}_i$ and user $\mathbf{k}_u$, another crucial problem is how to model their interaction based on the representation. Traditional shallow NCF models simply concatenate or sum user and item embedding for final prediction, assuming each dimension of the latent space is independent and linearly combined with the same weight for the prediction. In this section, we propose a new convolutional collaboration layer model by integrating the complex correlations between embedding dimensions into modelling. Specifically, we use the outer product above the latent user and item vector, and get a two-dimensional interaction map, which is more expressive and semantically reasonable. The target of this section is to estimate the matching score between user u and item i, (*i.e.,* $\hat{y}_{ui}$); and then we can generate a personalised recommendation list of items for a user based on the scores as shown in upper right in Fig. 3.

3.4.1. Interaction map

To obtain the complex correlation matrix, we propose to use an outer product operation 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 $S \times S$:

$$\mathbf{E} = \mathbf{k}_u \otimes \mathbf{j}_i = \mathbf{k}_u \mathbf{j}_i^T, \tag{21}$$

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

The idea of interaction map is a critical design to ensure the effectiveness of K-NCR for the recommendation. Compared to prior work (He et al., 2017; Zhang et al., 2017), 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.

3.4.2. Hidden CNN layers

Above the interaction map is a stack of hidden layers, which targets at extracting the useful signal from the interaction map. It is subjected to design and can be abstracted as $\mathbf{g} = f_\Theta(\mathbf{E})$, where f_Θ denotes the model of hidden layers that has parameters Θ, and $\mathbf{g}$ is the output vector to be used for the final prediction. Here we choose convolutional networks as the hidden layer, which stacks layers in a locally connected manner and utilises much fewer parameters than MLP. The similar technique was originally applied in recommendation for personalised ranking (He, Du et al., 2018). This allows us to build deeper models than MLP easily, and benefits the learning of high-order correlations among embedding dimensions.

Fig. 5 shows the model of hidden CNN layers. After getting the interaction map $\mathbf{E}$, we apply 32 kernels of size 2×2 filters to the input interaction map (with size 64×64). 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

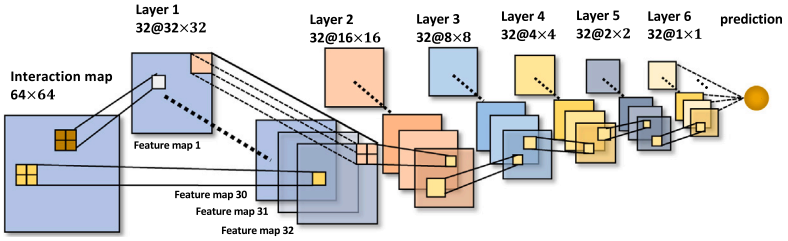


Fig. 5. Hidden CNN layers with embedding size 64.

Table 2
Basic statistics settings for the three datasets.

Dataset	MovieLens-20M	Book-Crossing	Last.FM
# users	114,571	18,390	1,872
# items	13,482	16,374	3,846
# interactions	10,653,587	143,630	42,346
# entities	93,323	31,227	10,828
# KG triples	397,698	61,175	16,753
# relations	32	18	60

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 obtain the final prediction score. We use the rectifier unit as the activation function, a common choice in CNN to build deep models. 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.

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

3.5. Learning algorithm

The complete loss function of our model is as follows:

$$\begin{aligned}\min \mathcal{L} =& \alpha \left[-\log\left(\sigma(\mathbf{z}_u^\top \mathbf{z}_v)\right) - \right. \\ & \left. Q \cdot \mathbb{E}_{v_n \sim P_n(v)} \log\left(\sigma(-\mathbf{z}_u^\top \mathbf{z}_{v_n})\right) \right] \\ & + \sum_{(u,i)\in D} \mathcal{J}(\hat{y}_{ui}, y_{ui})\end{aligned} \tag{22}$$

where the first part captures the KG embedding loss as defined in Eq. (9), and the second part models the prediction loss ($\mathcal{J}$ is the cross-entropy loss function). α denotes the balance parameter.

In our experiment all model parameters are trained in a joint manner, which is called joint training. Particularly, joint training optimises the parameters in the KG 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. Besides, we can also conduct multi-task learning by alternately calling optimiser for each part, where the propagating-based knowledge embedding is learnt first. Then we fine-tune neural collaborative filtering for the predicted rating with the fixed parameter in first part (Wu et al., 2018). Actually, the second manner works as the pre-training and fine tuning of the deep learning (LeCun et al., 2015). In this alternative approach, the rating prediction process entirely relies on the KG embedding part, while the KG embedding part could not benefit from rating information. We would show the superiority of joint training in the experimental part (see Table 2).

4. Experiment

4.1. Datasets

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

- **MovieLens-20M**[2] dataset contains almost 20 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, designed to explore implicit feedback, we transform explicit feedback of all three datasets into implicit one. Specifically, each implicit feedback is marked as 1, which denotes a positive rate, while we also randomly sample negative examples from the unknown items. Following previous efforts (Wang, Zhang, Zhao et al., 2019), the threshold of positive rating for MovieLens-20M 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] (Release 2016–10), we retrieve related triplet facts from DBpedia using SPARQL queries. We consider the mapping from item indices in the raw rating file to entity IDs in the KG. Then, we extract the triplets that are directly related to the entities aligned with items, no matter which role (i.e., subject or object) it serves as. Finally, we construct a new edge for each triplet connecting two entities. The basic statistics of the three datasets are presented in Table 2. 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 K-NCR 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 KG.

- **LibFM** (Rendle, 2012) 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.

[2] https://grouplens.org/datasets/movielens/20m/.
[3] http://www2.informatik.uni-freiburg.de/~cziegler/BX/.
[4] https://grouplens.org/datasets/hetrec-2011/.
[5] https://wiki.dbpedia.org/.

Table 3
The results of AUC and $F1$ in CTR prediction.

Model	MovieLens-20M		Book-Crossing		Last.FM	
	AUC	F1	AUC	F1	AUC	F1
LibFM	0.921 (−5.3%)	0.878 (−5.7%)	0.629 (−14.0%)	0.598 (−16.0%)	0.694 (−8.6%)	0.648 (−7.7%)
LibFM+TransE	0.928 (−4.6%)	0.889 (−4.5%)	0.649 (−11.2%)	0.613 (−13.9%)	0.709 (−6.6%)	0.652 (−7.1%)
NCF	0.925 (−4.9%)	0.884 (−5.0%)	0.642 (−12.2%)	0.597 (−16.2%)	0.704 (−7.2%)	0.646 (−8.0%)
PER	0.794 (−18.4%)	0.753 (−19.1%)	0.593 (−18.9%)	0.577 (−19.0%)	0.603 (−20.6%)	0.568 (−19.1%)
CKE	0.887 (−8.8%)	0.842 (−9.6%)	0.622 (−14.9%)	0.608 (−14.6%)	0.684 (−9.9%)	0.643 (−8.4%)
W&D	0.932 (−4.2%)	0.891 (−4.3%)	0.684 (−6.4%)	0.645 (−9.4%)	0.716 (−5.7%)	0.654 (−6.8%)
RippleNet	0.947 (−2.7%)	0.896 (−3.8%)	0.697 (−4.7%)	0.651 (−8.6%)	0.724 (−4.6%)	0.675 (−3.8%)
KGCN	0.955 (−1.8%)	0.913 (−1.9%)	0.705 (−3.6%)	0.684 (−3.9%)	0.736 (−3.0%)	0.687 (−2.1%)
NCR	0.923 (−5.1%)	0.887 (−4.7%)	0.650 (−11.1%)	0.598 (−16.0%)	0.695 (−8.4%)	0.664 (−5.4%)
KCR	0.952 (−2.2%)	0.894 (−4.0%)	0.694 (−5.1%)	0.654 (−8.1%)	0.729 (−4.0%)	0.682 (−2.8%)
K-NCR_A	0.961 (−1.2%)	0.918 (−1.4%)	0.704 (−3.7%)	0.678 (−4.8%)	0.738 (−2.8%)	0.684 (−2.6%)
K-NCR_J	**0.973**	**0.931**	**0.731**	**0.712**	**0.759**	**0.702**

- **LibFM + TransE**. We concatenate the raw features of users and items, as well as the corresponding averaged entity embeddings learned from TransE (Bordes et al., 2013) as input for LibFM. The dimension of TransE is 32.
- **NCF** (He et al., 2017) 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.
- **CKE** (Zhang et al., 2016) 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 32.
- **PER** (Yu et al., 2014) 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. Specifically, we manually design "user-item-attribute-item" meta-path as similarity feature.
- **Wide&Deep** (Cheng et al., 2016) 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 dimension of 100 and 50 as well as a wide channel.
- **RippleNet** (Wang, Zhang, Wang et al., 2018) is a knowledge graph embedding-based method that propagates and aggregates user potential preference on the KG for the recommendation.
- **KGCN** (Wang, Zhao et al., 2019) 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. Specifically, we use variant KGCN-sum as our baseline.

Besides, to examine the effectiveness of incorporated knowledge graph, we prepare two simplified variants of K-NCR. **KCR** does not use interaction map and attention based user preference modelling in the collaboration layer. **NCR** is another variant that does not incorporate the Knowledge Graph for the recommendation. We also use two training methods for K-NCR, a joint training approach named K-NCR_J and an alternative training approach as K-NCR_A.

4.3. Experiment setup

In K-NCR, we set the dimension of user and item latent vector as $S = 64$, and the number of CNN hidden layers is set as 6. The dropout rate is set as $\rho = 0.2$ for default, which are determined by optimising AUC on a validation set. For the knowledge propagation item modelling, we set $ReLU$ for non-last-layer aggregator, and $tanh$ for last-layer aggregator. We use open-source implementation of baselines to obtain performance result in all three datasets.

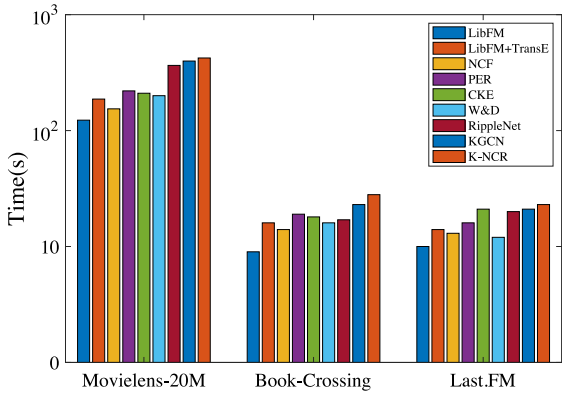


Fig. 6. Computational cost comparison.

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 evaluate our method in click-through rate (CTR) prediction scenarios. For CTR recommendation, the threshold of positive rating is set to 0.5. If $\hat{y}_{ui} >= 0.5$, we will recommend item i to user u. We apply the trained model to each piece of interactions in the test set and output the predicted click probability. Average AUC (Area under the ROC Curve) and $F1$ on test sets are used as the evaluation criterion for CTR prediction. Statistically significant improvement by unpaired two-sample t-test is with p = 0.05.

4.4. Overall performance

4.4.1. Comparison with baselines

The results of all methods in CTR prediction is presented in Table 3. The : computational cost comparison results are shown in Fig. 6. We have the following observations:

- In all cases, our proposed K-NCR outperform all the baseline methods on all the datasets. In particular, K-NCR improves over the strongest baselines *w.r.t* AUC by 1.8%, 3.6%, and 3.0% in MovieLens-20M, Book-Crossing, and Last-FM, respectively. By

stacking multiple relation-aware attentive embedding propagation layers, K-NCR is capable of exploring the high-order connectivity 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 indicates that the hand-crafted meta-paths cannot 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 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 and KGCN work well among these baselines. Both RippleNet and KGCN are Structure-based methods that use breadth-first-search to obtain multi-hop neighbours of an entity in the KG. RippleNet adopts a similar 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 is a representative of inward aggregation that learns the representation of an entity by aggregating information from its multi-hop neighbours to mine users' potential preferences. However, both RippleNet and KGCN can hardly capture the complex pairs interaction between users and items.
- Training takes up significantly more time compared to testing. Thus, we only report the training time here. It can be observed that simpler methods take less training time. LibFM is most efficient in terms of training, while KG-based methods all require the longer training time. Nevertheless, our method is still very efficient, and can be trained in less than seven minutes on the largest dataset Movielens-20M.

4.4.2. Comparison with model variants

We compare K-NCR with our proposed two simplifications. We can see from Table 3 that removing knowledge graph embedding and attention components degrades the model's performance. Specifically, KCR justifies the effectiveness of the attention mechanism and interaction map in interaction matching function modelling, specifying the attentive weights w.r.t. compositional items. Also, incorporating the knowledge graph embedding could further alleviate the sparsity problem and improve prediction performance in K-NCR. Besides, NCR (without KG) achieve better performance than NCF (Neural Collaborative Filtering). Because the proposal of using outer product above the embedding layer results in a two-dimensional interaction map that is more expressive and semantically plausible, which can learn high-order correlations among embedding dimensions. When comparing the two training variants of proposed K-NCR with different training strategy, K-NCR_J consistently achieve better performance than K-NCR_A with about 1.2% to 5% improvement. This experimental result shows that the joint training methods can exploit shared features across two domains (user–item interaction and extra KG), which can mutually benefit both the KG embedding part and the NCR part.

4.4.3. Results in sparse scenarios

One major advantage of incorporating knowledge graph in K-NCR 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-20M 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-20M dataset is that MovieLens-20M 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 4.

Table 4
Results of AUC on MovieLens-20M in CTR prediction with different ratios of training set r.

Model	r				
	20%	40%	60%	80%	100%
LibFM	0.725	0.814	0.846	0.892	0.921
LibFM+TransE	0.773	0.834	0.856	0.904	0.928
NCF	0.757	0.816	0.847	0.882	0.925
PER	0.638	0.693	0.724	0.769	0.794
CKE	0.725	0.784	0.831	0.857	0.887
W&D	0.769	0.839	0.876	0.914	0.932
RippleNet	0.852	0.873	0.886	0.922	0.947
KGCN	0.863	0.881	0.893	0.928	0.955
K-NCR_J	**0.904**	**0.934**	**0.951**	**0.962**	**0.973**

With the reducing of the training set, we observe that the performance of all methods deteriorate. When $r = 20\%$, the AUC score decreases by 21.3%, 16.7.2%, 18.2%, 19.6%, 18.3%, 17.5%, 10.0% and 9.6% for the baselines compared with the case when full training set is used ($r = 100\%$). In contrast, the AUC score of K-NCR only decreases by 7.1%, which shows that K-NCR can still maintain a decent performance even when the user–item interaction is sparse.

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 parameter α and (3) the impact of KG propagating layers.

4.5.1. Impact of feature map number

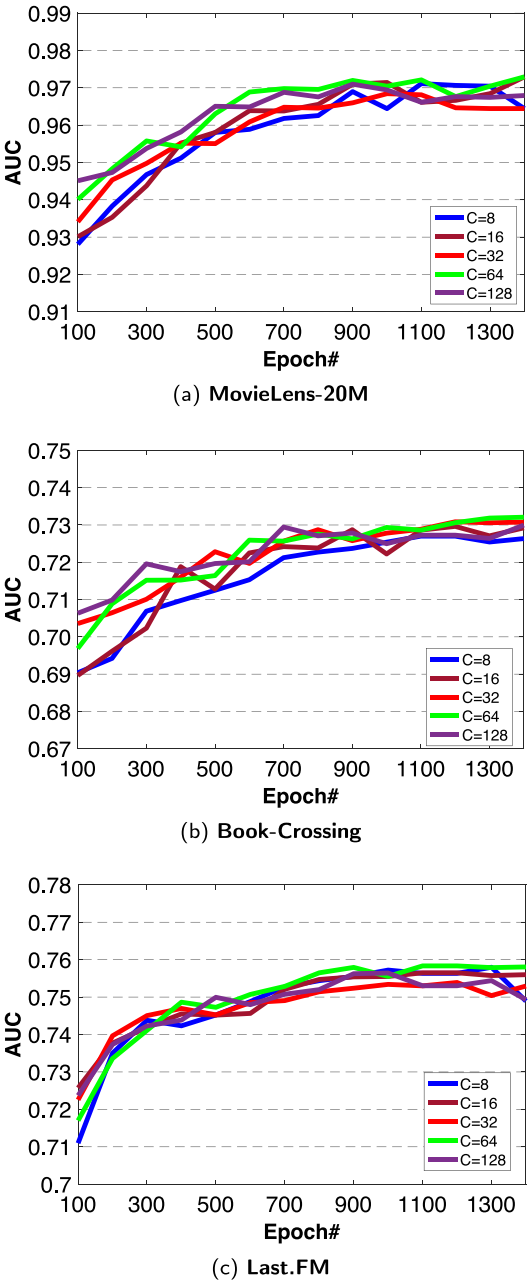
The number of feature maps in each CNN layer decides the dimension of final prediction layer, and will finally affect the representation ability of our K-NCR. Fig. 7 shows the performance of K-NCR with respect to different numbers of feature maps. We can see that although there are some slight differences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

4.5.2. Impact of tradeoff parameter

There is an important balance parameter α in the final loss function (22), which decides the weights between the knowledge graph embedding loss and the prediction loss. The larger value of α will put more weight on the KG embedding loss. We show the results with the varying parameter α in Fig. 8. For K-NCR_A, we first learn the KG embedding and then incorporate the KG embedding in neural collaborative part, which makes it not sensitive to α. For K-NCR_J, 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 $\alpha = 0$, K-NCR_J degenerates to the NCR without any knowledge graph information. When we increase α from 0, we actually strengthen the KG embedding for item modelling. However, a too large value of α would drive the objective function bias to the KG embedding learning. Given the experimental finding, we set $\alpha = 0.15$ for all three datasets.

4.5.3. Impact of KG propagating layers

We also vary the size of local knowledge graph context neighbourhoods by the maximal layer number L and observe the performance changes. The results are listed in Table 5, which clearly shows that an L with 1 or 2 is enough for real cases. Specifically, the relative smaller scale of Last.FM data need only one layer of neighbours to achieve best performance. We attribute this phenomenon to the trade-off between the useful information from long-distance dependency neighbours and

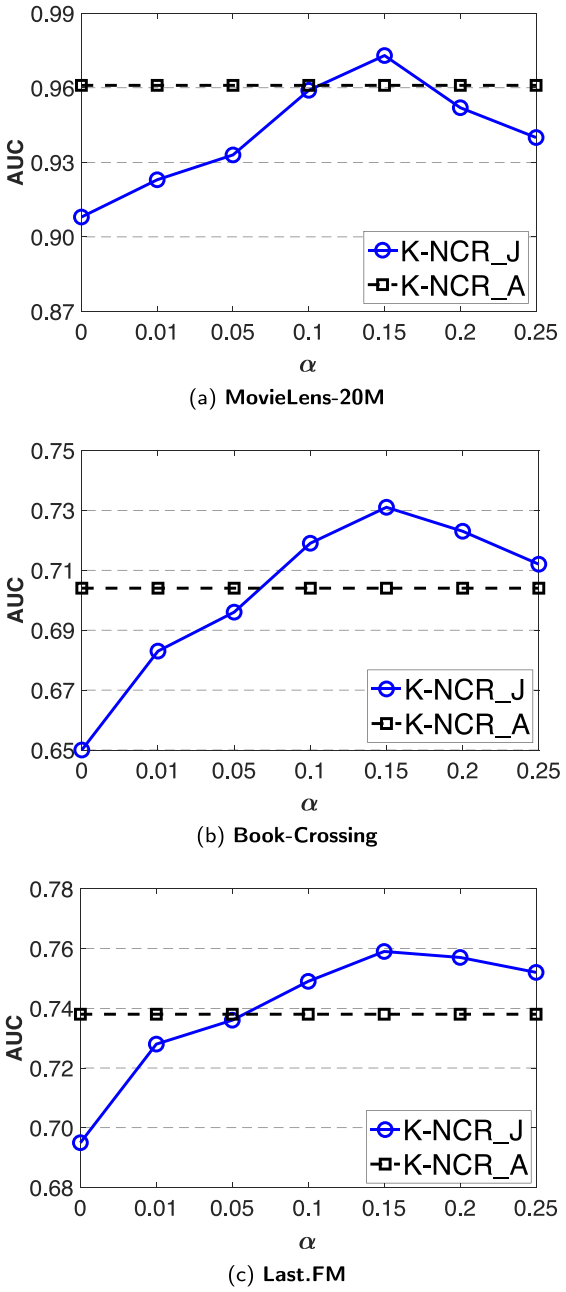


(a) MovieLens-20M

(b) Book-Crossing

(c) Last.FM

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



(a) MovieLens-20M

(b) Book-Crossing

(c) Last.FM

Fig. 8. Impact of parameter α.

useless information from noise: (1) a too-small layer value L is unable to adequately explore deep interdependency and gain enough context information, which is carried by the second- or third-order connectivities in larger scale graphs; (2) a too-large value of L may lead to overfitting problem as the number of parameters increases, while at the same time bringing in more undesired noise from neighbours. When the number of layers is $L = 1$, representation of each entity is a mixture of itself and its immediate neighbours. The experiment results indicate that the discrimination of the nodes is decreasing as the number of layers increases. We suggest that the increasing layers makes the neighbours of nodes more similar, and further makes node representations more similar or had a low degree of differentiation. The neighbour size grows exponentially with an increase of L-layers, so might introduce noise, and consequently decrease accuracy. For smaller size graph, the neighbour sets are easier to overlap with each other than that in large graph. We investigate how the scale of knowledge graph affects the choice of propagating layer. We sample a subset dataset of MovieLens-20M and Book-Crossing to test our hypothesis. The results are shown in Tables 6 and 7.

Table 5
Effect of embedding propagation layer numbers (L).

	MovieLens-20M		Book-Crossing		Last.FM	
	AUC	F1	AUC	F1	AUC	F1
K-NCR-1	0.892	0.872	0.693	0.683	**0.759**	**0.702**
K-NCR-2	**0.973**	**0.931**	**0.731**	**0.712**	0.743	0.683
K-NCR-3	0.968	0.909	0.725	0.706	0.647	0.589
K-NCR-4	0.726	0.653	0.524	0.593	0.424	0.448

Table 6
Basic statistics settings for the sampled datasets.

Dataset	Subset of MovieLens-20M	Subset of Book-Crossing
# users	1,761	2,179
# items	3,482	1,265
# interactions	72,939	32,432
# entities	21,624	10,976
# KG triples	45,653	18,465
# relations	32	18

Table 7
Effect of embedding propagation layer numbers (L).

	Subset of MovieLens-20M		Subset of Book-Crossing	
	AUC	F1	AUC	F1
K-NCR-1	**0.741**	**0.658**	**0.637**	**0.628**
K-NCR-2	0.702	0.623	0.618	0.604
K-NCR-3	0.643	0.603	0.542	0.537
K-NCR-4	0.608	0.544	0.515	0.528

5. Related work

5.1. Knowledge graph embedding

The Knowledge Aggregating-based Item Modelling module in K-NCR links to a large number of emerged work in KGE methods. KGE is used to embed entities or relations in a KG into continuous vector spaces, so that KG can be easier to process while still preserving the inherent structural information (Wang, Mao et al., 2017). In general, existing KGE methods can be classified into two categories: (1) Translation-based embedding methods try to model the relations between entities as translations from head entity to tail entity by exploiting distance-based loss functions, such as TransE (Bordes et al., 2013), TransH (Wang et al., 2014), and TransR (Lin et al., 2015); (2) Semantic matching methods try to model plausibility of entity pairs by exploiting similarity-based loss function between low-dimension latent semantics vectors of entities and relations, such as ANALOGY (Nickel et al., 2016), RESCAL (Nickel et al., 2011) and HolE (Liu et al., 2017). To enrich the KGE semantic information, some recent works are proposed to learn KGE by making full use of auxiliary data, such as textual attributes (Zhong et al., 2015) and entity types (Xie et al., 2016). However, the above KGE methods can hardly mine high-order potential interactions among entities, which is crucial to exploit users' potential interests to a long-range distance.

5.2. Graph convolutional network

Recently, Graph Neural Networks (Scarselli et al., 2008) was introduced to apply neural networks in the graph structure data. GCN (Kipf & Welling, 2017) extend CNN to aggregate information from long-term dependency graph structure context, which has received increasing research attention. There are two streams of GCN construction: spectral GCN (Defferrard et al., 2016; Kipf & Welling, 2017) and spatial GCN (Boscaini et al., 2016; Chen et al., 2018; Monti et al., 2017). Spectral GCN is based on the spectrum of graph Laplacian. Features of the graph are firstly transformed using Fourier transform and the convolutions are performed in the spectral domain. Spatial GCN approaches define convolutions directly on general graphs, operating on spatially close neighbours. Our method falls into this category, which offers us a chance to perform embedding propagation (or message passing) along the graph structure to faraway neighbour context. At its core is the neighbourhood aggregation, which iteratively aggregates the representations of each node with that of its adjacent nodes as its new representations. Inspired by the idea, researchers attempt to leverage GNNs to model high-order connectivity in recommender systems (Sang et al., 2020; Wu et al., 2018; Ying et al., 2018). For example, PinSage (Ying et al., 2018) employs the graph convolutional networks (Hamilton et al., 2017) to the pin-board bipartite graph in Pinterest and then update the representation of each pin. Wu et al. (2018) use GCNs on user/item intrinsic structure graphs to learn user/item representations. The difference between these works and ours is that they are all designed for homogeneous bipartite graphs or user/item similarity graphs where GCNs can be used directly, while here we investigate GCNs for heterogeneous KGs.

5.3. Recommendations with knowledge graphs

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

5.3.1. Embedding-based methods

Prior efforts (Wang, Wang et al., 2018; Wang, Zhang, Wang et al., 2018; Wang, Zhang, Xie et al., 2018; Zhang et al., 2016) 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 (Zhang et al., 2016) and DKN (Wang, Zhang, Xie et al., 2018) generate semantic embeddings from KG via knowledge graph embedding (KGE) techniques, and then incorporate them into recommenders — matrix factorisation (MF) (Rendle et al., 2009) and neural MF (He et al., 2017), respectively. RippleNet (Wang, Zhang, Wang et al., 2018) and KGCN (Wang, Zhao et al., 2019) apply a graph neural network to compute personalised item embeddings, which can capture item relationship by mining their associated attributes on the KG for recommendation. Recently, KGNN-LS (Wang, Zhang, Zhang et al., 2019) provides extra regularisation over the edge weights for recommendation by adding label smoothness to KGCN method.

5.3.2. Path-based methods

Another line of research introduces meta-paths (Hu, Shi et al., 2018; Yu et al., 2013; Zhao et al., 2017) and paths (Sun et al., 2018; Wang, Wang et al., 2018) to directly infer user preferences. For instance, FMG (Zhao et al., 2017) and MCRec (Hu, Shi et al., 2018) encode paths as embeddings to represent the user–item connectivity w.r.t meta-paths; RippleNet (Wang, Zhang, Wang et al., 2018) 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 labour-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.

5.4. Attention mechanism

Our work is also related to the attention-based models. Rather than using all available information equally, attention mechanism aims to focus on the most relevant parts of the input to make decisions. One of the benefits of attention mechanisms is that they allow for dealing with variable sized inputs and has been applied to various tasks, including machine translation (Bahdanau et al., 2015; Luong et al., 2015), sentence summarisation (Rush et al., 2015; Shen et al., 2018) and question answering (Hermann et al., 2015). For recommendation task, the attention mechanism is introduced to provide the ability to refer specific records dynamically in the decoder. NAIS (He, He et al., 2018), propose a neural network model named Neural Attentive Item Similarity model (NAIS) for item-based CF. For the user modelling, the proposed K-NCR adopt similar attention mechanism to distinguish which historical items in a user profile are more important for a prediction. For item modelling, the proposed K-NCR use additional knowledge graph to learn the representation of item entity, which

can provide richer semantics **than simple** one-hot embedding used in NAIS. Besides, the user and item representation are then fed into a newly designed two-dimensional interaction map with convolutional hidden layers to model the complex pairwise correlations between their embedding dimensions explicitly.

5.5. Neural collaborative filtering

There are two types of methods for implementing an effective neural collaborative filtering model (Deng et al., 2019; He, Du et al., 2018; He et al., 2017). One is based on representation learning and the other one is based on matching function learning.

5.5.1. Collaborative filtering based on representation learning

Since early work Funk-SVD (Funk, 2006) win the Netflix Prize competition, matrix factorisation based methods have become the mainstream of recommendation research over the past ten years (Hu et al., 2014; Koren et al., 2009; Ma, 2013; Salakhutdinov & Mnih, 2008). 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 (Sedhain et al., 2015) is the first model that try to use autoencoder to learn user and item representation. DMF (Xue et al., 2017) propose a 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 representation 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 the matching score.

5.5.2. Collaborative filtering based on matching function learning

Neural collaborative filtering(NCF) (He et al., 2017) 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 (Bai et al., 2017) is a variant of NCF that takes user neighbours and item neighbours as inputs. To better capture pairwise correlations user and item dimension, ConvNCF (He, Du et al., 2018) 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 (Cheng et al., 2016) 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 the auxiliary knowledge graph.

6. Conclusion

In this paper, we develop an end-to-end neural architecture K-NCR that jointly incorporates the knowledge graph structure and user–item interaction in a unified neural network model for recommendation. Specifically, we propose a propagation based knowledge graph embedding model that exploits the high-order structural proximity among entities in KG to enhance the item embedding learning in a unified framework. Besides, an attention-based user modelling part is proposed to learn the varying importance weights of the interacted items. We also design an interaction map based collaboration layer to model the complex user–item interaction behaviour in recommender systems. All the parameters in K-NCR 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.

Acknowledgements

This work has been supported by the National Key Research and Development Program of China under grant 2016YFB1000901, the China Scholarship Council (CSC), the National Natural Science Foundation of China under grant 91746209 and the Program for Changjiang Scholars and Innovative Research Team in University (PCSIRT) of the Ministry of Education of China under grant IRT17R32.

References

Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *3rd international conference on learning representations*.

Bai, T., Wen, J.-R., Zhang, J., & Zhao, W. X. (2017). A neural collaborative filtering model with interaction-based neighborhood. In *Proceedings of the 2017 ACM on conference on information and knowledge management* (pp. 1979–1982).

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., & Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems* (pp. 2787–2795).

Boscaini, D., Masci, J., Rodolà, E., & Bronstein, M. (2016). Learning shape correspondence with anisotropic convolutional neural networks. In *Advances in neural information processing systems* (pp. 3189–3197).

Cao, Y., Wang, X., He, X., Hu, Z., & Chua, T.-S. (2019). Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *The world wide web conference* (pp. 151–161). ACM.

Chen, X., Li, L.-J., Fei-Fei, L., & Gupta, A. (2018). Iterative visual reasoning beyond convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 7239–7248).

Cheng, H., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., Anderson, G., Corrado, G., Chai, W., Ispir, M., Anil, R., Haque, Z., Hong, L., Jain, V., Liu, X., & Shah, H. (2016). Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems, DLRS@RecSys 2016* (pp. 7–10).

Defferrard, M., Bresson, X., & Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems* (pp. 3844–3852).

Deng, Z., Huang, L., Wang, C., Lai, J., & Yu, P. S. (2019). DeepCF: A unified framework of representation learning and matching function learning in recommender system. In *The thirty-third AAAI conference on artificial intelligence* (pp. 61–68).

Funk, S. (2006). Netflix update: Try this at home. https://sifter.org/simon/journal/20061211.html, (Accessed 12 June 2019).

Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. In *Advances in neural information processing systems* (pp. 1024–1034).

He, X., Du, X., Wang, X., Tian, F., Tang, J., & Chua, T. (2018). Outer product-based neural collaborative filtering. In *Proceedings of the twenty-seventh international joint conference on artificial intelligence* (pp. 2227–2233).

He, X., He, Z., Song, J., Liu, Z., Jiang, Y.-G., & Chua, T.-S. (2018). NAIS: Neural attentive item similarity model for recommendation. *IEEE Transactions on Knowledge and Data Engineering*, *30*(12), 2354–2366.

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T.-S. (2017). Neural collaborative filtering. In *International conference on world wide web* (pp. 173–182).

Hermann, K. M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., & Blunsom, P. (2015). Teaching machines to read and comprehend. In *The annual conference on neural information processing systems, NeurIPS*.

Hu, B., Shi, C., Zhao, W. X., & Yu, P. S. (2018). 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* (pp. 1531–1540).

Hu, L., Sun, A., & Liu, Y. (2014). 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* (pp. 345–354).

Hu, G., Zhang, Y., & Yang, Q. (2018). CoNet: Collaborative cross networks for cross-domain recommendation. In *Proceedings of the 27th ACM international conference on information and knowledge management* (pp. 667–676).

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *International conference on learning representations*.

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, (8), 30–37.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436.

Lin, Y., Liu, Z., Sun, M., Liu, Y., & Zhu, X. (2015). Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on artificial intelligence*.

Liu, H., Wu, Y., & Yang, Y. (2017). Analogical inference for multi-relational embeddings. In *Proceedings of the 34th international conference on machine learning-Volume 70* (pp. 2168–2178). JMLR.org.

Luo, L., Xie, H., Rao, Y., & Wang, F. L. (2019). Personalized recommendation by matrix co-factorization with tags and time information. *Expert Systems with Applications*, *119*, 311–321.

Luong, M.-T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025.

Ma, H. (2013). 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).

Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., & Bronstein, M. M. (2017). Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 5115–5124).

Mun, J., Cho, M., & Han, B. (2017). Text-guided attention model for image captioning. In *Thirty-first AAAI conference on artificial intelligence*.

Nickel, M., Rosasco, L., & Poggio, T. A. (2016). Holographic embeddings of knowledge graphs. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence* (pp. 1955–1961).

Nickel, M., Tresp, V., & Kriegel, H.-P. (2011). A three-way model for collective learning on multi-relational data. In *ICML, vol. 11* (pp. 809–816).

Peng, Y., Fang, Y., Xie, Z., & Zhou, G. (2019). Topic-enhanced emotional conversation generation with attention mechanism. *Knowledge-Based Systems*, *163*, 429–437.

Qian, S., Zhang, T., & Xu, C. (2018). Cross-domain collaborative learning via discriminative nonparametric Bayesian model. *IEEE Transactions on Multimedia*, *20*(8), 2086–2099.

Rendle, S. (2012). Factorization machines with libFM. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *3*(3), 57:1–57:22.

Rendle, S., Freudenthaler, C., Gantner, Z., & Schmidt-Thieme, L. (2009). BPR: Bayesian personalized ranking from implicit feedback. In *UAI 2009, Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence* (pp. 452–461).

Rush, A. M., Chopra, S., & Weston, J. (2015). A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 conference on empirical methods in natural language processing* (pp. 379–389).

Salakhutdinov, R., & Mnih, A. (2008). Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proceedings of the 25th international conference on machine learning* (pp. 880–887).

Sang, L., Xu, M., Qian, S., Martin, M., Li, P., & Wu, X. (2020). Context-dependent propagating based video recommendation in multimodal heterogeneous information networks. *IEEE Transactions on Multimedia*.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2008). The graph neural network model. *IEEE Transactions on Neural Networks*, *20*(1), 61–80.

Schlichtkrull, M., Kipf, T. N., Bloem, P., Van Den Berg, R., Titov, I., & Welling, M. (2018). Modeling relational data with graph convolutional networks. In *European semantic web conference* (pp. 593–607). Springer.

Sedhain, S., Menon, A. K., Sanner, S., & Xie, L. (2015). Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th international conference on world wide web* (pp. 111–112).

Shen, T., Zhou, T., Long, G., Jiang, J., Pan, S., & Zhang, C. (2018). DiSAN: Directional self-attention network for RNN/CNN-free language understanding. In *Proceedings of the thirty-second AAAI conference on artificial intelligence* (pp. 5446–5455).

Sourabh, V., & Chowdary, C. (2019). Peer recommendation in dynamic attributed graphs. *Expert Systems with Applications*, *120*, 335–345.

Sun, Z., Wu, B., Wu, Y., & Ye, Y. (2019). APL: Adversarial pairwise learning for recommender systems. *Expert Systems with Applications*, *118*, 573–584.

Sun, Z., Yang, J., Zhang, J., Bozzon, A., Huang, L.-K., & Xu, C. (2018). Recurrent knowledge graph embedding for effective recommendation. In *Proceedings of the 12th ACM conference on recommender systems* (pp. 297–305).

Symeonidis, P., & Malakoudis, D. (2019). Multi-modal matrix factorization with side information for recommending massive open online courses. *Expert Systems with Applications*, *118*, 261–271.

Tay, Y., Anh Tuan, L., & Hui, S. C. (2018). Latent relational metric learning via memory-based attention for collaborative ranking. In *Proceedings of the 2018 world wide web conference* (pp. 729–739). International World Wide Web Conferences Steering Committee.

Wang, C.-D., Deng, Z.-H., Lai, J.-H., & Philip, S. Y. (2018). Serendipitous recommendation in e-commerce using innovator-based collaborative filtering. *IEEE Transactions on Cybernetics*, (99), 1–15.

Wang, X., He, X., Nie, L., & Chua, T.-S. (2017). 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* (pp. 185–194).

Wang, Q., Mao, Z., Wang, B., & Guo, L. (2017). Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, *29*(12), 2724–2743.

Wang, X., Wang, D., Xu, C., He, X., Cao, Y., & Chua, T.-S. (2018). Explainable reasoning over knowledge graphs for recommendation. arXiv preprint arXiv:1811.04540.

Wang, X., Wang, D., Xu, C., He, X., Cao, Y., & Chua, T. (2019). Explainable reasoning over knowledge graphs for recommendation. In *The thirty-third AAAI conference on artificial intelligence* (pp. 5329–5336).

Wang, Z., Zhang, J., Feng, J., & Chen, Z. (2014). Knowledge graph and text jointly embedding. In *Proceedings of the 2014 conference on empirical methods in natural language processing* (pp. 1591–1601).

Wang, H., Zhang, F., Wang, J., Zhao, M., Li, W., Xie, X., & Guo, M. (2018). Ripple network: Propagating user preferences on the knowledge graph for recommender systems. In *The 27th ACM international conference on information and knowledge management* (pp. 7–18).

Wang, H., Zhang, F., Xie, X., & Guo, M. (2018). DKN: Deep knowledge-aware network for news recommendation. In *Proceedings of the 2018 world wide web conference on world wide web* (pp. 1835–1844).

Wang, H., Zhang, F., Zhang, M., Leskovec, J., Zhao, M., Li, W., & Wang, Z. (2019). Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 968–977).

Wang, H., Zhang, F., Zhao, M., Li, W., Xie, X., & Guo, M. (2019). Multi-task feature learning for knowledge graph enhanced recommendation. arXiv preprint arXiv:1901.08907.

Wang, H., Zhao, M., Xie, X., Li, W., & Guo, M. (2019). Knowledge graph convolutional networks for recommender systems. In *The world wide web conference* (pp. 3307–3313).

Wu, Y., Liu, H., & Yang, Y. (2018). Graph convolutional matrix completion for bipartite edge prediction. In *International joint conference on knowledge discovery, knowledge engineering and knowledge management* (pp. 51–60).

Wu, L., Sun, P., Hong, R., Ge, Y., & Wang, M. (2018). Collaborative neural social recommendation. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 1–13.

Xie, R., Liu, Z., & Sun, M. (2016). Representation learning of knowledge graphs with hierarchical types. In *Proceedings of the 25th international joint conference on artificial intelligence* (pp. 2965–2971).

Xiong, R., Wang, J., Zhang, N., & Ma, Y. (2018). Deep hybrid collaborative filtering for web service recommendation. *Expert Systems with Applications*, *110*, 191–205.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., & Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning* (pp. 2048–2057).

Xue, H.-J., Dai, X.-Y., Zhang, J., Huang, S., & Chen, J. (2017). Deep matrix factorization models for recommender systems. In *Proceedings of the 26th international joint conference on artificial intelligence* (pp. 3203–3209).

Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., & Leskovec, J. (2018). Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International conference on knowledge discovery & data mining* (pp. 974–983).

Yu, X., Ren, X., Gu, Q., Sun, Y., & Han, J. (2013). Collaborative filtering with entity similarity regularization in heterogeneous information networks. In *International joint conferences on artificial intelligence, vol. 27*. Citeseer.

Yu, X., Ren, X., Sun, Y., Gu, Q., Sturt, B., Khandelwal, U., Norick, B., & Han, J. (2014). 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).

Zhang, Y., Ai, Q., Chen, X., & Croft, W. (2017). Joint representation learning for top-n recommendation with heterogeneous information sources. In *Proceedings of the 2017 ACM on conference on information and knowledge management* (pp. 1449–1458).

Zhang, Y., Lai, G., Zhang, M., Zhang, Y., Liu, Y., & Ma, S. (2014). Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *Proceedings of the 37th international ACM SIGIR conference on research & development in information retrieval* (pp. 83–92).

Zhang, F., Yuan, N. J., Lian, D., Xie, X., & Ma, W.-Y. (2016). 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).

Zhao, H., Yao, Q., Li, J., Song, Y., & Lee, D. L. (2017). 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).

Zheng, E., Kondo, G. Y., Zilora, S., & Yu, Q. (2018). Tag-aware dynamic music recommendation. *Expert Systems with Applications*, *106*, 244–251.

Zhong, H., Zhang, J., Wang, Z., Wan, H., & Chen, Z. (2015). Aligning knowledge and text embeddings by entity descriptions. In *Proceedings of the 2015 conference on empirical methods in natural language processing* (pp. 267–272).