mathematics

MDPI

Article

Collaborative Knowledge-Enhanced Recommendation with Self-Supervisions

Zhiqiang Pan and Honghui Chen *

Science and Technology on Information Systems Engineering Laboratory,
National University of Defense Technology, Changsha 410073, China; panzhiqiang@nudt.edu.cn
* Correspondence: chenhonghui@nudt.edu.cn; Tel.: +86-135-1731-8075

Abstract: Knowledge-enhanced recommendation (KER) aims to integrate the knowledge graph (KG) into collaborative filtering (CF) for alleviating the sparsity and cold start problems. The state-of-the-art graph neural network (GNN)–based methods mainly focus on exploiting the connectivity between entities in the knowledge graph, while neglecting the interaction relation between items reflected in the user-item interactions. Moreover, the widely adopted BPR loss for model optimization fails to provide sufficient supervisions for learning discriminative representation of users and items. To address these issues, we propose the collaborative knowledge-enhanced recommendation (CKER) method. Specifically, CKER proposes a collaborative graph convolution network (CGCN) to learn the user and item representations from the connection between items in the constructed interaction graph and the connectivity between entities in the knowledge graph. Moreover, we introduce the self-supervised learning to maximize the mutual information between the interaction- and knowledge-aware user preferences by deriving additional supervision signals. We conduct comprehensive experiments on two benchmark datasets, namely Amazon-Book and Last-FM, and the experimental results show that CKER can outperform the state-of-the-art baselines in terms of recall and NDCG on knowledge-enhanced recommendation.

Keywords: recommender systems; knowledge-enhanced recommendation; collaborative filtering; knowledge graph; self-supervised learning

Citation: Zhiqiang, P.; Honghui, C. Collaborative Knowledge-Enhanced Recommendation with Self-Supervisions. *Mathematics* **2021**, *9*, 2129. https://doi.org/10.3390/math9172129

Academic Editors: Ezequiel López-Rubio and Alessandro Niccolai

Received: 2 July 2021
Accepted: 30 August 2021
Published: 2 September 2021

1. Introduction

Recommender systems (RS) are an effective method to filter the irrelevant information on the internet and maintain a user's personalized needs [1–3], which has wide applications, such as search engines [4], e-commerce websites [5], etc. As a classical recommendation method, CF aims to generate recommendations by learning user and item representations from the user–item interactions [6–8]. However, CF faces the data sparsity issue and cold start problem (that is, making recommendations for novel users who have no preference for any items, or recommending new items that have not been interacted with by any users) [9–11], leading to unsatisfactory performance. To address these limitations, KER is proposed to introduce the knowledge graph (KG) into CF for enriching the connectivity between items in scenarios, where the knowledge graph can be accessed, thus learning high-quality latent vector for users and items to enhance recommendations.

Earlier methods for knowledge-enhanced recommendation can be divided into the embedding-based methods [12–15] and path-based methods [16–20], where the embedding-based methods learn the user and item representations, using the knowledge base, and the path-based methods enhance the recommendation by exploiting the connectivity pattern between items in the KG. For instance, Zhang et al. propose to enhance the item representations with the structural, textual and visual knowledge extracted from the knowledge base [12], and Wang et al. propose an end-to-end framework to perform explicit reasoning on KG to learn path semantics and improve the recommendation interpretability [18].

Recently, the GNN-based methods were proposed by adopting the GNNs to unify the aforementioned two categories of methods [9,10,21,22]. For example, Wang et al. propose to detect the user interest by iteratively propagating the user preference in the KG [9], and Wang et al. further exploit the high-order connectivity among users, items and entities in the constructed collaborative knowledge graph [10].

Though the above-mentioned methods have achieved considerable performance, there still remains several limitations. First, the embedding- and path-based methods show unsatisfactory performance since they either leverage the semantic representation of items or model the semantic connectivity between items, without taking both aspects of information into consideration. Moreover, the state-of-the-art GNN-based recommenders mainly focus on exploiting the high-order connectivity between entities in the knowledge graph, neglecting the interaction relation (i.e., interacted by the same user) between items. In addition, the widely adopted Bayesian personalized ranking (BPR) loss cannot provide sufficient supervision signals to learn an accurate representation of users and items, failing to effectively distinguish the candidate items and accurately rank them when making predictions.

To address the above issues, we propose the CKER method. Specifically, given the historical user–item interactions, we first construct an interaction graph to reflect the interaction relation between items. Then, we propose the CGCN for learning the user and item representations, including two channels which exploit the connection between items in the interaction graph and the semantic connectivity between entities in the knowledge graph, respectively. After that, we construct a user–item bipartite graph from the historical interactions, which is utilized to generate the interaction- and knowledge-aware user preferences by relying on the generated representation of user's interacted items in two convolution channels. Next, we derive the self-supervisions by adopting the information noise-contrastive estimation (InfoNCE) [23] to maximize the mutual information between the interaction- and knowledge-aware preferences of each user, which is jointly trained with the main supervised learning.

Comprehensive experiments are conducted on two benchmark datasets, namely Amazon-Book and Last-FM. The significant improvement of CKER above the state-of-the-art baselines in terms of recall and NDCG demonstrates the effectiveness of our proposal.

We summarize the main contributions in this paper as follows:

1. To the best of our knowledge, we are the first to simultaneously consider the connection between items reflected in the user–item interactions and the semantic connectivity between entities in the knowledge graph;
2. We introduce the self-supervised learning to derive additional supervision signals for enhancing the representation learning of users and items by maximizing the mutual information between the user's interaction- and knowledge-aware preferences;
3. Extensive experiments conducted on two benchmark datasets, namely Amazon-Book and Last-FM, demonstrate the superiority of CKER over the competitive baselines in terms of recall and NDCG on the knowledge-enhanced recommendation task.

The rest of this paper is organized as follows. First, the related literature is summarized in Section 2. Then, we detail our proposed CKER model in Section 3. After that, we describe the experimental settings in Section 4 and analyze the experimental results in detail in Section 5. Finally, we conclude this work and suggest our future directions in Section 6.

2. Related Work

In this section, we first review the previous work for the knowledge-enhanced recommendation in Section 2.1, and then summarize the related work about the self-supervised learning and its applications in RS in Section 2.2.

2.1. Knowledge-Enhanced Recommendation

The existing work for knowledge-enhanced recommendation can be mainly divided into three categories, i.e., the embedding-based methods, the path-based methods and

the GNN-based methods. The embedding-based methods generally utilize the semantics connections in KG to enhance the representation learning of items or users. For example, Cao et al. propose to perform the multi-task learning by combining the item recommendation and the knowledge graph completion to introduce the item knowledge into the user preference generation [14]. Moreover, the path-based models aim to exploit the connectivity pattern between items in the KG for guiding the personalized recommendation. For instance, Yu et al. regard KG as a heterogeneous information network and diffuse the user preferences along different meta-paths to generate the latent vector of users and items [19]. In addition, Wang et al. design a knowledge-aware path recurrent network to capture the sequential dependencies of entities and relations on each path in KG for detecting the user intent [18]. Upon the embedding- and path-based methods, the GNN-based methods are proposed to simultaneously consider the semantic representation of items and the connectivity pattern between items in KG. For example, Wang et al. propose to detect the user interest by propagating embeddings over the entities related to user's interacted items in the knowledge graph [9]. Then, Wang et al. propose KGCN, which learns user-specific item embeddings by exploiting the high-order connectivity between entities in the KG [21], and then Wang et al. further extends KGCN by adding regularization over the edge weights in KG using the label smoothness to alleviate the overfitting problem [22]. In addition, Wang et al. propose KGAT to integrate both the user–item interactions and the item knowledge into a hybrid collaborative knowledge graph for modeling the high-order relations among users, items and entities [10].

However, the existing knowledge-based recommenders fail to take both the interaction relation between items reflected in the user–item interactions and the connectivity pattern between items contained in the knowledge graph into consideration simultaneously, leading to unsatisfactory recommendation performance.

2.2. Self-Supervised Learning

Self-supervised learning (SSL) aims to train a network and learns the data representations by deriving the supervision signals from the raw data automatically, which can be categorized into the generative methods [24–26] and the contrastive methods [23,27–29]. The generative models aim to reconstruct the input data, where the popular methods include the variational autoencoder (VAE) [30,31], the generative adversarial networks (GAN) [32], etc. Differently, the constrastive models learn data representations by comparing a training sample from different views based on the noise contrastive estimation (NCE) [33], such as from the global–local views [34] or the global–global views [27]. In this work, we adopt the contrastive method for deriving the self-supervisions, which can avoid introducing additional parameters.

Considering the effectiveness of SSL in various fields, recently SSL was also introduced into RS for enhancing the recommendation [35–40]. For example, Yao et al. propose to apply the SSL in the large-scale recommendation scenario to solve the data sparse and long-tail problems [35]. Moreover, Wu et al. propose to develop the SSL in the graph-based collaborative filtering by comparing the user and item embeddings from different augmentation views to learn discriminative representations [36]. Furthermore, Yu et al. propose to adopt the SSL to maximize the mutual information among the user preferences modeled by different channels to improve the social recommendation [37]. In addition, Xia et al. design a dual channel hypergraph convolutional network for session-based recommendation, where the SSL is utilized to enhance the hypergraph modeling by contrasting the session representation generated from the local graph and global graph [38].

However, to the best of our knowledge, SSL has not been applied on the knowledge-enhanced recommendation task for improving the recommendation accuracy. Thus, in this paper, we introduce the SSL to derive the additional supervisions by comparing the user preferences generated from the interaction graph and the knowledge graph for enhancing the representation of users and items.

3. Approach

In this section, we first formulate the definition of the knowledge-enhanced recommendation task. Then, we describe our proposed CKER method in detail, which mainly consists of five components, namely the CGCN module, the user preference generation, the main supervised learning module, the self-supervised learning module and the multi-task learning module.

The framework of CKER is plotted in Figure 1. Given the user–item interactions, we first construct an interaction graph to establish the interaction relation between items. Then for each item, we design a CGCN to update the item representations by propagating information from its neighbors in the interaction graph and the knowledge graph, respectively. After that, the user preference is generated by combining the interaction- and knowledge-aware preferences modeled by relying on the learned item representations for making predictions. Next, we apply the Bayesian personalized ranking (BPR) loss as the main supervised loss to utilize the supervisions in the user-item interactions by reconstructing the interaction matrix. In addition, we introduce the self-supervised learning, which adopts the InfoNCE to derive the self-supervisions between the generated interaction- and knowledge-aware preferences for each user. Finally, the multi-task learning is conducted by combining both the supervised and self-supervised losses for model optimization.

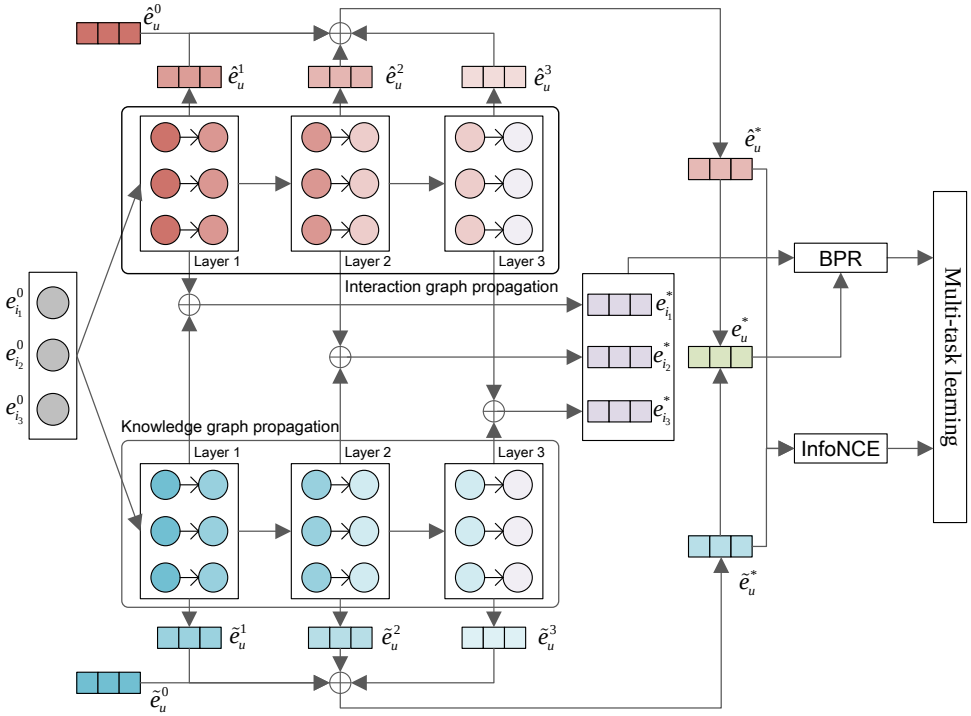


Figure 1. The framework of CKER.

Let $\mathcal{U}$ and $\mathcal{I}$ be the user set and item set, respectively, and the interactions between users and items are denoted as $\mathcal{O}^+$, where each user–item pair $(u,i) \in \mathcal{O}^+$ indicates that user u interacted with item i. Moreover, assuming the knowledge graph is $\mathcal{G}_K = \{(h,r,t)|h,t \in \mathcal{V}, r \in \mathcal{R}\}$, where $\mathcal{V}$ is a set of real-world entities and $\mathcal{R}$ is the relation set between entities. For example, (*Braveheart*, *director*, *Mel Gibson*) denotes that the movie *Braveheart* is directed by the director *Mel Gibson*, where *Braveheart* and *Mel Gibson* are the entities and "*is directed by*" denotes the relation between two entities. In addition, the items in the interactions are the subset of the entities in the KG, i.e., $\mathcal{I} \subset \mathcal{V}$. The aim of knowledge-enhanced recommendation is to learn the representation of users and items from the interaction data $\mathcal{O}^+$ and knowledge graph $\mathcal{G}_K$, so as to predict how likely each user is to adopt the candidate items; then, items ranked at the top-K positions constitute the recommendation list for the user.

The main notations used in this paper are listed in Table 1.

Table 1. Main notations used in this paper.

Notation	Description
$\mathcal{U}$	the user set consists of all users
$\mathcal{I}$	the item set consists of all items
$\mathcal{V}$	the entity set consists of all entities in KG
$\mathcal{R}$	the relation set consists of all relations in KG
$\mathcal{O}^+$	the observed interactions between $\mathcal{U}$ and $\mathcal{V}$
$\mathcal{G}_K$	the knowledge graph
$\mathcal{G}_I$	the constructed interaction graph
$\mathcal{G}_U$	the constructed user-item bipartite graph
$\mathcal{N}_i^K$	the neighbors of entity i in the knowledge graph
$\mathcal{N}_i^I$	the neighbors of item i in the interaction graph
$\mathcal{N}_u^U$	the neighbors of user u in the user-item bipartite graph
d	the dimension of user and item embeddings
$\hat{\mathbf{e}}_i^l$	the vector of item i at the l-layer IG convolution
$\tilde{\mathbf{e}}_i^l$	the vector of item i at the l-layer KG convolution
$\mathbf{e}_i^l$	the vector of item i at the l-layer CGCN
$\mathbf{e}_i^*$	the final representation of item i
$\hat{\mathbf{e}}_u^*$	the interaction-aware preference of user u
$\tilde{\mathbf{e}}_u^*$	the knowledge-aware preference of user u
$\mathbf{e}_u^*$	the final preference of user u
$\hat{y}_{ui}$	the prediction score of user u on item i
λ	a hyper-parameter in the InfoNCE
α	a hyper-parameter in the multi-task learning
$\mathcal{L}_{main}, \mathcal{L}_{ssl}, \mathcal{L}$	the main supervised, self-supervised and final loss
L	the layer number of graph convolutions
K	the length of the recommendation list for the user
M_I	the neighbor number of each node in the interaction graph
M_K	the neighbor number of each node in the knowledge graph

3.1. Collaborative Graph Convolution Networks

In order to simultaneously exploit the interaction relation between items reflected in the user-item interactions and the item knowledge introduced by the knowledge graph, we propose the CGCN to learn the item representations, which consists of two channels, i.e., the interaction graph propagation and the knowledge graph propagation.

3.1.1. Interaction Graph Propagation

First, we take the interaction relation between items reflected in the user–item interactions into consideration. Specifically, given the interactions between users and items as $\mathcal{O}^+$, we first construct an interaction graph (IG) as $\mathcal{G}_I = \{\mathcal{I}, \mathcal{E}_I\}$, where $\mathcal{I}$ denotes the nodes, i.e., all items, and $\mathcal{E}_I$ is the edges. Each edge $(i_\epsilon, i_\kappa) \in \mathcal{E}_I$ indicates that item i_ϵ and item i_κ are interacted by the same user. Moreover, we apply the max sampling according to the edge weights to select the M_I most related items as the final neighbors of each item, so as to filter out the noise introduced by user's uncertain behavior pattern.

After constructing the interaction graph, we propagate information from the neighbors of each item in $\mathcal{G}_I$ to exploit the interaction relation between items for updating the item representations. More specifically, we adopt the light graph convolution (LGC) proposed in [41] to conduct the information propagation. Differently, we adopt a left normalization method considering its simplicity and low computation cost, and the comparison of different normalization methods in CKER is left to our future work. Specifically, the l-layer graph convolution for item i can be formalized as follows:

$$\hat{\mathbf{e}}_i^l = \frac{1}{|\mathcal{N}_i^I|} \sum_{\epsilon \in \mathcal{N}_i^I} \hat{\mathbf{e}}_\epsilon^{l-1}, \tag{1}$$

where $\hat{\mathbf{e}}_i^l \in \mathbb{R}^d$ is the propagated information for item i at the l-layer interaction graph convolution, $\mathcal{N}_i^I$ is the neighbors of item i in the interaction graph and $|\mathcal{N}_i^I|$ is the neighbor number, and $\hat{\mathbf{e}}_\epsilon^{l-1} \in \mathbb{R}^d$ is the representation of neighbor $\epsilon \in \mathcal{N}_i^I$ at the $(l-1)$-layer convolution.

3.1.2. Knowledge Graph Propagation

Besides propagating information on the interaction graph, we utilize the knowledge graph convolution to exploit the connectivity between entities in the knowledge graph. More specifically, each triplet $(i, r, v) \in \mathcal{G}_K$ denotes that entity i and entity v are connected by the relation r. Moreover, similar to that in the IG, we apply the max sampling by relying on the edge weights to select the M_K most related entities as the final neighbors of each entity to avoid introducing bias. In the KG, each tail entity has different semantics when paired with different relations; for example, entity *Mel Gibson* plays the role as the *director* and *star* in two triplets (*Braveheart*, *director*, *Mel Gibson*) and (*Braveheart*, *star*, *Mel Gibson*), respectively. Thus, we obtain the neighbor information for each entity by aggregating its corresponding relation–tail pairs in the KG as follows:

$$\tilde{\mathbf{e}}_i^l = \frac{1}{|\mathcal{N}_i^K|} \sum_{(r,v)\in\mathcal{N}_i^K} \mathbf{e}_r \odot \tilde{\mathbf{e}}_v^{l-1}, \tag{2}$$

where $\tilde{\mathbf{e}}_i^l \in \mathbb{R}^d$ is the propagated information for entity i at the l-layer knowledge graph convolution, $\mathcal{N}_i^K$ is the neighbors of entity i in the knowledge graph. $\mathbf{e}_r$ is the relation vector generated by the embedding layer preceding the graph propagation architectures, $\tilde{\mathbf{e}}_v^{l-1} \in \mathbb{R}^d$ is the representation of entity $v \in \mathcal{N}_i^K$ at the $(l-1)$-layer convolution, and $\odot$ denotes the Hadamard product used for combination. For each triplet (i, r, v), we propagate the information from the tail v to the head i under the relation r by multiplying the latent vector of the relation and the tail in an element-wise way, so that the relational message can be carried together with the tail in the information propagation.

3.1.3. Multi-Layer Graph Convolutions

At the l-layer CGCN, after propagating information on the interaction graph and knowledge graph to generate the respective item representations, i.e., $\hat{\mathbf{e}}_i^l$ and $\tilde{\mathbf{e}}_i^l$, we combine them together by the sum pooling to obtain the final latent vector of item i:

$$\mathbf{e}_i^l = \hat{\mathbf{e}}_i^l + \tilde{\mathbf{e}}_i^l, \tag{3}$$

where $\mathbf{e}_i^l \in \mathbb{R}^d$ is the representation of item i generated by the l-layer CGCN.

Moreover, multi-layer CGCNs can be stacked to exploit multi-hop connection between items in IG and high-order connectivity between entities in KG. Specifically, for item i, the l-layer CGCN can be formalized as follows:

$$\mathbf{e}_i^l = \mathrm{CGCN}(\mathbf{e}_i^{l-1}, \mathcal{N}_i^I, \mathcal{N}_i^K), \tag{4}$$

where $\mathbf{e}_i^l$ and $\mathbf{e}_i^{l-1}$ are the respective representation of item i at the l- and $(l-1)$-layer CGCN, and $\mathcal{N}_i^I$ and $\mathcal{N}_i^K$ are the neighbors of item i in IG and KG, respectively.

After generating the representation of items at different CGCN layers, we obtain the final item representations by summing them together, which can be formalized as follows:

$$\mathbf{e}_i^* = \mathbf{e}_i^0 + \mathbf{e}_i^1 + \cdots + \mathbf{e}_i^L, \tag{5}$$

where $\mathbf{e}_i^* \in \mathbb{R}^d$ is the final representation of item i, and $\mathbf{e}_i^0 \in \mathbb{R}^d$ is initialized by the embedding layer.

3.2. User Preference Generation

After learning the item representations by multi-layer CGCNs, we generate the user preference by the latent vector of user's interacted items. More specifically, we construct a user-item bipartite graph as $\mathcal{G}_U = \{\mathcal{U} \cup \mathcal{I}, \mathcal{E}_U\}$ from the user–item interactions, where the nodes $\mathcal{U} \cup \mathcal{I}$ include all users and all items, and each edge $(u, i) \in \mathcal{E}_U$ indicates that user u interacted with item i before.

Given the generated item representations at the l-layer interaction graph convolution, we obtain the user preference reflected in the interaction graph as follows:

$$\hat{\mathbf{e}}_u^l = \frac{1}{|\mathcal{N}_u^U|} \sum_{i \in \mathcal{N}_u^U} \hat{\mathbf{e}}_i^{l-1}, \tag{6}$$

where $\hat{\mathbf{e}}_u^l \in \mathbb{R}^d$ denotes the user preference aggregated from the representation of user's interacted items at the $(l-1)$-layer interaction graph convolution, and $\mathcal{N}_u^U$ is the interacted items of user u.

Then, similar to that in Equation (5), we generate the interaction-aware user preference by summing the preference generated from different interaction graph convolution layers:

$$\hat{\mathbf{e}}_u^* = \hat{\mathbf{e}}_u^0 + \hat{\mathbf{e}}_u^1 + \cdots + \hat{\mathbf{e}}_u^L, \tag{7}$$

where $\hat{\mathbf{e}}_u^* \in \mathbb{R}^d$ is the interaction-aware preference of user u generated by combining the user interest at different interaction graph convolution layers, and $\hat{\mathbf{e}}_u^0 \in \mathbb{R}^d$ is initialized by the embedding layer before the graph convolutions.

Similarly, we can also obtain the knowledge-aware user preference by aggregating the representation of items generated by multi-layer knowledge graph convolutions, which can be formalized as follows:

$$\tilde{\mathbf{e}}_u^l = \frac{1}{|\mathcal{N}_u^U|} \sum_{i \in \mathcal{N}_u^U} \tilde{\mathbf{e}}_i^{l-1}, \tag{8}$$

$$\tilde{\mathbf{e}}_u^* = \tilde{\mathbf{e}}_u^0 + \tilde{\mathbf{e}}_u^1 + \cdots + \tilde{\mathbf{e}}_u^L, \tag{9}$$

where $\tilde{\mathbf{e}}_u^*$ is the knowledge-aware preference of user u obtained by summing the preference at different knowledge graph convolution layers, and $\tilde{\mathbf{e}}_u^0$ is generated by the embedding layer before the graph convolutions.

Next, we can obtain the final user preference $\mathbf{e}_u^*$ by combining the interaction- and knowledge-aware preferences together:

$$\mathbf{e}_u^* = \hat{\mathbf{e}}_u^* + \tilde{\mathbf{e}}_u^*, \tag{10}$$

3.3. Supervised Learning

After obtaining the item representations and generating the user preference, following [10], we adopt the inner product to make predictions as follows:

$$\hat{y}_{ui} = {\mathbf{e}_u^*}^\mathsf{T} \mathbf{e}_i^*, \tag{11}$$

where $\hat{y}_{ui}$ is the predicted score of measuring the probability of user u adopting item i.

Then, to learn the trainable parameters in CKER (i.e., the ID embeddings of users and items), following [10], the Bayesian personalized ranking (BPR) loss is adopted as the optimization objective to utilize the supervision signals in the user–item interactions. Specifically, the BPR loss encourages the target items to be ranked at the top positions by enlarging the distance between the prediction score of the ground truth and the negative sample as follows:

$$\mathcal{L}_{main} = \sum_{(u,i,j) \in \mathcal{O}} -\log(\sigma(\hat{y}_{ui} - \hat{y}_{uj})), \tag{12}$$

where $\mathcal{L}_{main}$ is the main supervised loss, $\mathcal{O} = \{(u,i,j)|(u,i) \in \mathcal{O}^+, (u,j) \in \mathcal{O}^-\}$ indicates the pairwise training data, σ denotes the sigmoid function, and item i is user's interacted item, while j is an item randomly sampled from the unobserved interactions, i.e., $\hat{y}_{uj} \in \mathcal{U} \times \mathcal{I} \backslash \mathcal{O}^+$.

3.4. Self-Supervised Learning

In order to maximize the mutual information between the user preference obtained from the interaction graph and the knowledge graph, we introduce the self-supervised learning to derive the self-supervision signals using the InfoNCE [23] for enhancing the representation of users and items. Specifically, for each user u, we obtain the interaction-aware and knowledge-aware preferences reflected in the IG and KG as $\hat{\mathbf{e}}_u^*$ and $\tilde{\mathbf{e}}_u^*$, respectively. Assuming that the current mini-batch consists of N users, then the knowledge-aware preference of user u_ϵ should be more similar to the interaction-aware preference of user u_ϵ than that of the other $N-1$ users in the mini-batch. Based on this intuition, we adopt the InfoNCE [23], which regards the pair of knowledge- and interaction-aware preferences of user u_ϵ (i.e., $\tilde{\mathbf{e}}_{u_\epsilon}^*$ and $\hat{\mathbf{e}}_{u_\epsilon}^*$) as the positive pair, and treats the pairs combining user u's knowledge-aware preference with the interaction-aware preference of other users in the mini-batch (i.e., $[(\tilde{\mathbf{e}}_{u_\epsilon}^*, \hat{\mathbf{e}}_{u_\kappa}^*)|\kappa = 1, \ldots, \epsilon-1, \epsilon+1, \ldots, N]$) as the negative samples. We formalize the InfoNCE as follows:

$$\mathcal{L}_{ssl} = \frac{\exp(\lambda \text{sim}(\tilde{\mathbf{e}}_{u_\epsilon}^*, \hat{\mathbf{e}}_{u_\epsilon}^*))}{\exp(\lambda \text{sim}(\tilde{\mathbf{e}}_{u_\epsilon}^*, \hat{\mathbf{e}}_{u_\epsilon}^*)) + \sum_{\kappa=1,\kappa\neq\epsilon}^{N} \exp(\lambda \text{sim}(\tilde{\mathbf{e}}_{u_\epsilon}^*, \hat{\mathbf{e}}_{u_\kappa}^*))}, \tag{13}$$

where $\mathcal{L}_{ssl}$ is the self-supervised loss, $\text{sim}(\mathbf{u},\mathbf{v}) = \cos(\mathbf{u},\mathbf{v}) = \mathbf{u}^\mathsf{T}\mathbf{v}/||\mathbf{u}||||\mathbf{v}||$ is the cosine similarity between two vectors $\mathbf{u}$ and $\mathbf{v}$, where $||\cdot||$ denotes the L2 normalization operation, and λ is a hyper-parameter for scaling the similarity. By introducing the additional supervisions using the InfoNCE, we can encourage the interaction-aware preference of different users to be uniformly distributed in the latent space, so as to learn discriminative embeddings of users and items for better distinguishing them when making predictions.

3.5. Multi-Task Learning

After obtaining the main supervised loss $\mathcal{L}_{main}$ by Equation (12) and the self-supervised loss $\mathcal{L}_{ssl}$ by Equation (13), we conduct the multi-task learning by combining them together:

$$\mathcal{L} = \mathcal{L}_{main} + \alpha \mathcal{L}_{ssl}, \tag{14}$$

where α is a hyper-parameter which adjusts the weight between two losses, a small $\mathcal{L}_{main}$ indicates that the target items are ranked at top positions and a small $\mathcal{L}_{ssl}$ denotes that the items are well distributed in the embedding space by the self-supervised learning. Finally, the back-propagation through time (BPTT) algorithm [42] is applied to optimize CKER for learning the trainable parameters.

We detail the learning procedure of CKER in Algorithm 1. Given the observed user–item interactions $\mathcal{O}^+$, we first construct the interaction graph and the user–item bipartite graph in lines 1 and 2, respectively. Then for each training mini-batch $B \in X$, where X denotes all mini-batches, we first conduct the information propagation by multi-layer CGCNs, which consist of the interaction and knowledge graph propagations from line 5 to 9, which are then fused to generate the final item representations in line 10. Next, for each user–item pair, we obtain the user preference from line 12 to 18, including generating the interaction- and knowledge-aware preferences. After that, we sample the negative item in line 19 and look up the representation of items in line 20, which are inputted into the prediction function together with the user preference to obtain the prediction scores in line 21. Then, we generate the main supervised loss in line 22 and the self-supervised loss in line 23, respectively. Finally, we obtain the multi-task training loss in line 25 and apply the back-propagation to optimize the model in line 26.

Algorithm 1 Learning algorithm of CKER.

Input: The observed interactions between users and items, i.e., $\mathcal{O}^+$;
The knowledge graph $\mathcal{G}_K = \{(h, r, t)|h, t \in \mathcal{V}, r \in \mathcal{R}\}$;
Output: The embeddings of users and items in CKER;

$\mathcal{G}_I = \{\mathcal{I}, \mathcal{E}_I\} \leftarrow$ GraphConstruct $(\mathcal{O}^+)$;
$\mathcal{G}_U = \{\mathcal{U} \cup \mathcal{I}, \mathcal{E}_U\} \leftarrow$ GraphConstruct $(\mathcal{O}^+)$;
for epoch in $1, \ldots,$ N$_{ep}$ **do**
  for minibatch $B \in X$ **do**
    for l in range(L) **do**
      $\hat{\mathbf{E}}^l =$ InteractionPropagation$(\hat{\mathbf{E}}^{l-1}, \mathcal{G}_I)$ based on Equation (1);
      $\tilde{\mathbf{E}}^l =$ KnowledgePropagation$(\tilde{\mathbf{E}}^{l-1}, \mathcal{G}_K)$ based on Equation (2);
      $\mathbf{E}^l = \hat{\mathbf{E}}^l + \tilde{\mathbf{E}}^l$ based on Equation (3);
    end for
    $\mathbf{E}^* =$ SumPooling$(\mathbf{E}^0, \mathbf{E}^1, \ldots, \mathbf{E}^L)$ based on Equation (5);
    for (u, i) in $\mathcal{O}_B^+$ **do**
      for l in range(L) **do**
        $\hat{\mathbf{e}}_u^l =$ User-ItemPropagation$(\hat{\mathbf{E}}^{l-1}, \mathcal{G}_U)$ based on Equation (6);
        $\tilde{\mathbf{e}}_u^l =$ User-ItemPropagation$(\tilde{\mathbf{E}}^{l-1}, \mathcal{G}_U)$ based on Equation (8);
      end for
      $\hat{\mathbf{e}}_u^* =$ SumPooling$(\hat{\mathbf{e}}_u^0, \hat{\mathbf{e}}_u^1, \ldots, \hat{\mathbf{e}}_u^L)$ based on Equation (7);
      $\tilde{\mathbf{e}}_u^* =$ SumPooling$(\tilde{\mathbf{e}}_u^0, \tilde{\mathbf{e}}_u^1, \ldots, \tilde{\mathbf{e}}_u^L)$ based on Equation (9);
      $\mathbf{e}_u^* = \hat{\mathbf{e}}_u^* + \tilde{\mathbf{e}}_u^*$ based on Equation (10);
      $j \leftarrow$ RandomSample(u);
      $\mathbf{e}_i^*, \mathbf{e}_j^* \leftarrow$ EmbeddingLookup$(\mathbf{E}^*, i, j)$;
      $\hat{y}_{ui}, \hat{y}_{uj} =$ Prediction$(\mathbf{e}_u^*, \mathbf{e}_i^*, \mathbf{e}_j^*)$ based on Equation (11);
      $\mathcal{L}_{main} =$ BPR$(\hat{y}_{ui}, \hat{y}_{uj})$ based on Equation (12);
      $\mathcal{L}_{ssl} =$ InfoNCE$(\tilde{\mathbf{e}}_u^*, \hat{\mathbf{e}}_u^*, \hat{\mathbf{e}}_\kappa^*), \kappa \in \mathcal{U}_B \backslash u$ based on Equation (13);
    end for
    Optimize joint learning loss: $\mathcal{L} = \mathcal{L}_{main} + \alpha \mathcal{L}_{ssl}$;
    use back-propagation to optimize the parameters;
  end for
end for
return User and item embeddings.

4. Experiments

4.1. Research Questions

We prove the effectiveness of CKER by addressing the following five research questions:

(RQ1) Can our proposed CKER achieve the state-of-the-art performance, compared with the baselines on the knowledge-enhanced recommendation?
(RQ2) How does each component in CKER contribute to the model performance?
(RQ3) How does the layer number of graph convolutions influence the recommendation accuracy?
(RQ4) How does CKER perform with different number of neighbors incorporated in the interaction and knowledge graphs?
(RQ5) What is the impact of the hyper-parameters α and λ on the performance of CKER?

4.2. Datasets and Evaluation Metrics

Two publicly available datasets, namely Amazon-Book and Last-FM, are adopted to evaluate the performance of CKER and the baselines. Amazon-Book is selected from Amazon-review, which is a widely used dataset for product recommendation. Last-FM is collected from the Last.fm online music systems, where the tracks are regarded as the items, and we take the subset of the music listening records from January 2015 to June 2015

for experiments as in [10]. Moreover, for both Amazon-Book and Last-FM, the 10-score setting [43] is applied to ensure the data quality, where users with fewer than 10 interactions and items appearing less than 10 times are filtered. In addition, following [10], for each dataset, we randomly select 80% of the historical interactions as the training set, and the remaining part constitutes the test set.

Moreover, the item knowledge is constructed for each dataset. Specifically, items are mapped to the Freebase [44,45] entities via title matching if there is mapping available, where we consider the triplets that are directly related to the entities aligned with items, no matter which role (i.e., subject or object) it serves. Moreover, two-hop neighbor entities of items are taken into consideration to enrich the relations between entities. Here, introducing small hops of neighbors can merely incorporate limited connectivities between items in the KG into the user and item representation learning, while introducing large hops of neighbors easily brings in bias. In this paper, we follow the setting in [10], which takes two-hop neighbor entities into consideration. We would like to leave the investigation on the impact of the hop number of neighbors to our future work. In addition, in order to ensure the KG quality, inactive entities (i.e., appearing less than 10 times) and infrequent relations (i.e., appearing in less than 50 triplets) are filtered out in the KG data for both Amazon-Book and Last-FM. The statistics of Amazon-Book and Last-FM after processing are shown in Table 2.

Table 2. Dataset statistics.

	Dataset	**Amazon-Book**	**Last-FM**
User-Item Interactions	#Users	70,679	23,566
	#Items	24,915	48,123
	#Interactions	847,733	3,034,796
Knowledge Graph	#Entities	88,572	58,266
	#Relations	39	9
	#Triplets	2,557,746	464,567

Following previous work [10], Recall@K and NDCG@K are adopted as the evaluation metrics to evaluate the recommendation performance. Recall@K measures whether the target items are contained in the top-K positions in the recommendation list, while NDCG@K takes the ranking of the target items into consideration, i.e., whether the recommender ranks the target items at right positions. Unless specified differently, K is set to 20 in our experiments.

4.3. Model Summary

To validate the effectiveness of CKER, we compare our proposed CKER with the following state-of-the-art baselines: (1) A KG-free method, i.e., MF [46]; (2) An embedding-based method, i.e., CKE [12]; and (3) Three GNN-based methods, i.e., KGNN-LS [22], KGAT [10] and CKAN [47].

- **MF** [46] does not take the KG into consideration and learns the user and item representations by reconstructing the interaction matrix using the matrix factorization, where users and items are simply represented by their ID embeddings;
- **CKE** [12] is an embedding-based method learning the entity embeddings using the knowledge base by TransR [48], which are then combined with the ID embeddings of items generated by MF as the final item representations for item predictions;
- **KGNN-LS** [22] computes personalized item embeddings, using GNNs on user-specific graphs transformed from the KG, and provides regularization over the edge weights, using the label smoothness to prevent overfitting;
- **KGAT** [10] combines the user–item graph and KG as a holistic graph for modeling the collaborative information by exploiting the high-order connectivity among users, items and entities. Moreover, the importance of different neighbors is distinguished

in an attentive way by learning discriminative representation of the interaction relationship and KG relations;
- **CKAN** [47] explicitly encodes the collaborative signals in user–item interactions, which are then combined with the knowledge associations modeled by an attention mechanism to discriminate the contribution of different neighbors in KG.

4.4. Experimental Setup

The hyper-parameters are tuned on the validation set, which is randomly separated from the interactions in the training set with a proportion of 10%. For a fair comparison, following [10], for both two datasets, we set the embedding dimension and the batch size to 64 and 1024, respectively, and ADAM [49] is adopted as the optimizer. Then, a grid search is conducted to confirm the optimal parameter settings on each dataset. More specifically, the learning rate ρ and L2 regularization η are respectively tuned in $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$ and $\{10^{-6}, 10^{-5}, \ldots, 10^{-2}\}$, the layer number of CGCNs L is ranged in $\{1, 2, 3\}$, the neighbor number in IG and KG, i.e., M_I and M_K, are both searched in $\{2, 4, 8, 16, 32\}$, and the parameters α and λ are tuned in $\{0.005, 0.01, 0.05, 0.1, 0.5\}$ and $\{6, 8, 10, 12, 14\}$, respectively. In addition, the model parameters, i.e., the ID embeddings of users and items are initialized with the Xavier [50] method. The best performing parameters on two datasets are summarized in Table 3.

Table 3. Hyper-parameter settings of CKER.

Dataset	ρ	η	L	M_I	M_K	α	λ
Amazon-Book	10^{-4}	10^{-5}	1	4	16	0.05	12
Last-FM	10^{-4}	10^{-5}	1	8	16	0.5	10

5. Results and Discussion

5.1. Overall Performance

The performance of our proposed CKER and the baselines are presented in Table 4. Here, the results of all baselines are directly taken from [10] since we adopt the same datasets and preprocessing method for the experiments. First, we can observe that the KG-free method MF performs worse than other baselines for all cases on two datasets, indicating the necessity of introducing the knowledge graph for enhancing recommendation. Moreover, compared to MF, we can see that CKE achieves slightly better performance, indicating the effectiveness of learning collaborative embeddings for items from the knowledge base.

Table 4. Model performance. The results of the best performing baseline and the best performer in each column are underlined and boldfaced, respectively. ▲ denotes a significant improvement of CKER over the best baseline, using a paired t-test ($p < 0.01$).

Method	Amazon-Book		Last-FM	
	Recall@20	NDCG@20	Recall@20	NDCG@20
MF	0.1300	0.0678	0.0724	0.0617
CKE	0.1342	0.0698	0.0732	0.0630
KGNN-LS	0.1362	0.0560	0.0870	0.0642
KGAT	<u>0.1487</u>	<u>0.0799</u>	<u>0.0873</u>	<u>0.0744</u>
CKAN	0.1442	0.0698	0.0812	0.0660
CKER	**0.1619** ▲	**0.0863** ▲	**0.0951** ▲	**0.0832** ▲
%Improv.	8.88%	8.01%	8.93%	11.83%

Moreover, we can see that the GNN-based methods can obviously outperform CKE and MF in terms of both Recall@20 and NDCG@20 on two datasets, which indicates the utility of exploiting the high-order connectivity of items the knowledge graph. Furthermore,

by comparing CKAN to KGNN-LS, we can observe that CKAN generally performs better than KGNN-LS, except losing the competition in terms of Recall@20 on Last-FM. We analyze that the possible reason is that, besides modeling the item knowledge, CKAN further takes the collaborative signals in the user–item interactions into consideration for making recommendations. In addition, by exploring the multi-order connectivity among users, items and entities in an attentive way in the collaborative knowledge graph, KGAT performs best among the baselines in terms of both metrics on two datasets.

Next, we move to the performance of our proposed CKER. First, it can be observed that CKER achieves the best performance in terms of both Recall@20 and NDCG@20 on two datasets. We attribute the improvements to the fact that (1) CKER can simultaneously exploit the interaction relation between items in the user–item interactions and the connectivity between entities in the knowledge graph; (2) CKER introduces the self-supervised learning to derive the self-supervision signals for enhancing the representation learning of users and items, which can help obviously distinguish different items on the basis of the main supervised learning. In addition, we can observe that CKER improves the performance above the best baseline KGAT by 8.88% and 8.01% in terms of Recall@20 and NDCG@20 on Amazon-Book, respectively, where the improvement rate is larger on Recall@20. However, the phenomenon is different on Last-FM, where a higher improvement rate is observed on the NDCG@20 metric (i.e., 11.83%) than that on Recall@20 (i.e., 8.93%). This could be due to the fact that the number of interactions and KG triplets on two datasets are different, which means that CKER contributes relatively more to hitting the target items in the recommendation list in scenarios where the item knowledge is more abundant than the interactions, while CKER improves the ability of ranking the target items at right positions more obviously in scenarios where the user–item interactions are relatively more sufficient.

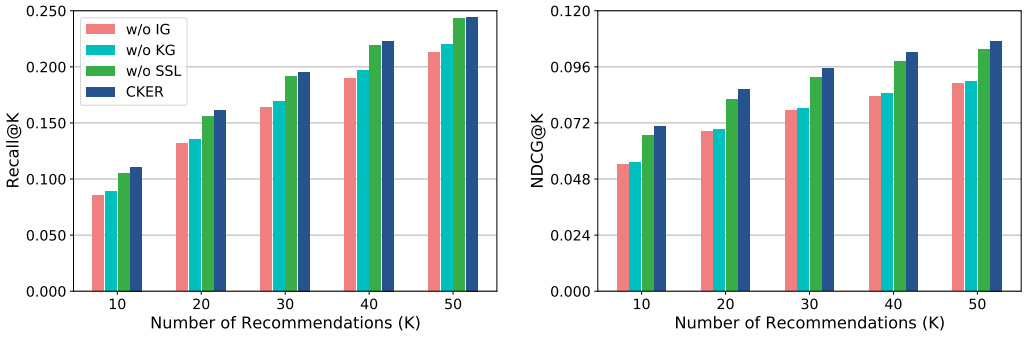
5.2. Ablation Study

For RQ2, in order to validate the utility of each component in CKER, we conduct an ablation study by comparing CKER with the following variants:

- **w/o IG** removes the interaction graph propagation and learns the representation of users and items by exploiting only the knowledge graph;
- **w/o KG** removes the knowledge graph propagation and generates the user and item representations by merely exploiting the interaction graph;
- **w/o SSL** removes the self-supervisions between the interaction- and knowledge-aware user preferences and optimizes the model, merely using the BPR loss.

We present the results of CKER and its variants in Figure 2, where we evaluate the model performance on Amazon-Book by ranging the recommendation number from 10 to 50 for providing a comprehensive comparison. Similar phenomena can also be observed on the Last-FM dataset.

From Figure 2, we can observe that removing each component from CKER consistently decreases the recommendation accuracy. Moreover, by comparing w/o SSL to w/o IG and w/o KG, we can observe that w/o SSL achieves obviously better performance than w/o IG and w/o KG, indicating the effectiveness of our proposed CGCN, which simultaneously exploits the interaction relation between items in IG and the connectivity between entities in KG. Furthermore, compared with w/o IG, we can see that w/o KG achieves better performance in terms of both Recall@K and NDCG@K on various K. We analyze that this may be due to that, compared to the interaction graph, the knowledge graph contains noise entities that are unrelated to the personalized recommendation, introducing much bias in modeling the user preference. In addition, it is observed that the performance gap between CKER and w/o SSL is decreasing when the recommendation number increases, especially on the Recall@K metric. This indicates that introducing the self-supervised learning contributes to the performance improving in short recommendation lists more obviously, which is practical in real-world applications since the interface for displaying the recommendation results may be limited, such as on mobile phones.

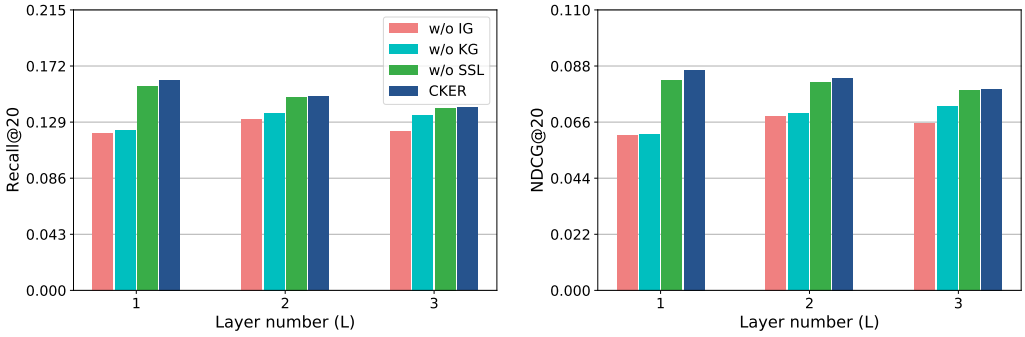


(**a**) Recall@K on Amazon-Book. (**b**) NDCG@K on Amazon-Book.

Figure 2. Ablation study.

5.3. Impact of Layer Number

For RQ3, to investigate the impact of the layer number of graph convolutions on the model performance, we compare CKER with its variants by ranging the layer number from 1 to 3. The results on Amazon-Book are provided in Figure 3, similar phenomena can also be observed on the Last-FM dataset.



(**a**) Recall@20 on Amazon-Book. (**b**) NDCG@20 on Amazon-Book.

Figure 3. Impact of graph convolution layer number.

From Figure 3, we can observe that for the variants w/o IG and w/o KG, with the layer number increasing, the performance of both variants consistently increases. This is due to that merely utilizing the KG propagation or the IG propagation, the model fails to provide sufficiently enough connectivities between items for learning the representations. Moreover, the best performance of both w/o IG and w/o KG achieved at layer number $L = 3$ is stills worse than w/o SSL and CKER. However, different from w/o IG and w/o KG, it is observed that with the layer number increasing, the performance of the variant w/o SSL and CKER both decreases. This may be due to the fact that by exploiting both the interaction and knowledge graphs, the recommender easily leads to overfitting, due to the abundant connectivities between items. Moreover, we can see that the performance gap between w/o SSL and CKER is decreasing with the layer number increasing. This could be explained by the fact that a larger number of graph convolutions leads to a more serious overfitting problem, decreasing the contribution of the self-supervised learning to the performance improving.

5.4. Impact of Neighbor Number

For RQ4, in order to investigate the impact of the neighbor number in the interaction and knowledge graphs on the model performance, we apply a grid search by ranging the neighbor number in IG (denoted as M_I) and the neighbor number in KG (denoted as M_K) both in $\{2, 4, 8, 16, 32\}$. The results on Amazon-Book and Last-FM are presented in Tables 5 and 6, respectively.

From Table 5, we can observe that on Amazon-Book, the best performance in terms of both metrics is achieved at $M_K = 32$ and $M_I = 2$, where the neighbor number in the IG is obviously smaller than that in the KG. This may be due to the fact that the IG has a more powerful ability to exploit the connectivity between items compared to the KG since unrelated entities to the recommendation task may be introduced in KG as stated in Section 5.2. Thus, a relatively smaller M_I is required for achieving the best performance. Moreover, for the Last-FM dataset, the best performance of CKER in terms of both Recall@20 and NDCG@20 is achieved at $M_K = 8$ and $M_I = 8$. Compared to the results on Amazon-Book, we can see that for the neighbor numbers achieving the best performance, M_K decreases and M_I increases. We attribute this difference to the fact that the number of interactions and triplets in two datasets are different as shown in Table 2. Since fewer triplets are contained in Last-FM than Amazon-Book, M_K achieving the best performance on Last-FM is also correspondingly smaller than that on Amazon-Book. Similarly, the larger number of interactions in Last-FM than Amazon-Book explains the larger M_I achieving the best performance on the Last-FM dataset.

Table 5. Impact of the neighbor number for Amazon-Book.

			Recall@20		
	$M_I = 2$	$M_I = 4$	$M_I = 8$	$M_I = 16$	$M_I = 32$
$M_K = 2$	0.1326	0.1315	0.1269	0.1241	0.1175
$M_K = 4$	0.1421	0.1411	0.1372	0.1331	0.1294
$M_K = 8$	0.1547	0.1545	0.1525	0.1485	0.1445
$M_K = 16$	0.1583	0.1614	0.1595	0.1547	0.1525
$M_K = 32$	**0.1618**	0.1610	0.1593	0.1568	0.1538
			NDCG@20		
$M_K = 2$	0.0724	0.0711	0.0677	0.0655	0.0611
$M_K = 4$	0.0766	0.0756	0.0730	0.0699	0.0676
$M_K = 8$	0.0833	0.0828	0.0808	0.0785	0.0762
M_K =16	0.0843	0.0863	0.0844	0.0809	0.0797
$M_K = 32$	**0.0869**	0.0860	0.0847	0.0827	0.0809

Table 6. Impact of the neighbor number for Last-FM.

			Recall@20		
	$M_I = 2$	$M_I = 4$	$M_I = 8$	$M_I = 16$	$M_I = 32$
$M_K = 2$	0.0914	0.0917	0.0926	0.0904	0.0863
$M_K = 4$	0.0919	0.0934	0.0937	0.0913	0.0877
$M_K = 8$	0.0936	0.0939	**0.0955**	0.0926	0.0899
$M_K = 16$	0.0926	0.0939	0.0951	0.0931	0.0902
$M_K = 32$	0.0933	0.0938	0.0947	0.0936	0.0906
			NDCG@20		
$M_K = 2$	0.0793	0.0793	0.0809	0.0776	0.0733
$M_K = 4$	0.0801	0.0817	0.0828	0.0794	0.0764
$M_K = 8$	0.0816	0.0829	**0.0841**	0.0814	0.0778
$M_K = 16$	0.0815	0.0823	0.0834	0.0808	0.0771
$M_K = 32$	0.0822	0.0830	0.0832	0.0818	0.0788

5.5. Hyper-Parameter Analysis

For RQ5, to investigate the impact of the hyper-parameters α and λ on the performance of CKER, we perform a grid search by tuning α and λ in $\{0.005, 0.01, 0.05, 0.1, 0.5\}$ and $\{6, 8, 10, 12, 14\}$, respectively. The results on Amazon-Book are presented in Table 7, and the results on the Last-FM dataset are shown in Table 8.

For the parameter α, we can see that for each λ, increasing α will generally decrease the performance on Amazon-Book, while increasing the performance on Last-FM. We

analyze that the possible reason is that a larger α indicates that the intensity of the self-supervisions is larger, which can distinguish the items more obviously. As shown in Table 2, the item number in Last-FM is obviously larger than that in Amazon-Book, thus a larger α is required in the Last-FM dataset to learn discriminative item representations. Moreover, as for the parameter λ, it is observed that for each α on both Amazon-Book and Last-FM, with λ increasing, the performance of CKER generally first increases and then decreases. This may be due to the fact that the hyper-parameter λ plays a role of controlling the intensity of mining the hard negative samples [23,51], where a larger λ makes the learned item embeddings more uniformly distributed in the embedding space. Thus, with λ increasing from 6 to 14, CKER first learns more accurate representation of items, and then faces the overfitting problem, which degrades the recommendation accuracy.

Table 7. Impact of the hyper-parameters α and λ for Amazon-Book.

	Recall@20				
	$\lambda = 6$	$\lambda = 8$	$\lambda = 10$	$\lambda = 12$	$\lambda = 14$
$\alpha = 0.005$	0.1593	0.1607	0.1614	**0.1619**	0.1611
$\alpha = 0.01$	0.1589	0.1595	0.1600	0.1615	0.1594
$\alpha = 0.05$	0.1562	0.1562	0.1584	0.1600	0.1604
$\alpha = 0.1$	0.1545	0.1536	0.1561	0.1588	0.1601
$\alpha = 0.5$	0.1484	0.1465	0.1487	0.1529	0.1571
	NDCG@20				
$\alpha = 0.005$	0.0852	0.0860	0.0862	0.0863	0.0856
$\alpha = 0.01$	0.0847	0.0854	0.0855	0.0865	0.0844
$\alpha = 0.05$	0.0853	0.0857	0.0863	**0.0866**	0.0862
$\alpha = 0.1$	0.0854	0.0853	0.0859	0.0865	**0.0866**
$\alpha = 0.5$	0.0836	0.0829	0.0836	0.0847	0.0864

Table 8. Impact of the hyper-parameters α and λ for Last-FM.

	Recall@20				
	$\lambda = 6$	$\lambda = 8$	$\lambda = 10$	$\lambda = 12$	$\lambda = 14$
$\alpha = 0.005$	0.0907	0.0916	0.0919	0.0914	0.0908
$\alpha = 0.01$	0.0902	0.0911	0.0921	0.0919	0.0920
$\alpha = 0.05$	0.0899	0.0911	0.0919	0.0916	0.0910
$\alpha = 0.1$	0.0907	0.0920	0.0921	0.0925	0.0918
$\alpha = 0.5$	0.0936	0.0941	0.0939	**0.0942**	0.0940
	NDCG@20				
$\alpha = 0.005$	0.0783	0.0791	0.0794	0.0788	0.0783
$\alpha = 0.01$	0.0780	0.0790	0.0797	0.0795	0.0793
$\alpha = 0.05$	0.0788	0.0800	0.0806	0.0806	0.0797
$\alpha = 0.1$	0.0797	0.0807	0.0814	0.0812	0.0806
$\alpha = 0.5$	0.0817	0.0821	0.0823	**0.0827**	0.0824

6. Conclusions and Future Work

In this paper, we propose a novel approach, i.e., the CKER method. First, CKER designs a CGCN to simultaneously exploit the interaction relation between items reflected in the user–item interactions and the item knowledge in the knowledge graph. Moreover, we apply the self-supervised learning to derive additional supervision signals for distinguishing items by contrasting the user preferences generated from the interaction and knowledge graphs. Extensive experiments conducted on two benchmark datasets, namely Amazon-Book and Last-FM, validate that CKER can outperform the state-of-the-art baselines on the knowledge-enhanced recommendation task, achieving the improvements of 8.88–8.93% in terms of Recall@20 and 8.01–11.83% in terms of NDCG@20, respectively. However, for the

scenarios where the knowledge graph is unable or hard to construct, the advantages of our proposed CKER method may not be noticeable, leading to unsatisfactory performance.

For future work, we would like to incorporate various sources of side information, such as social networks, as knowledge for enhancing the recommendation. Moreover, we are also interested in improving the applicability of KG for recommendation by automatically filtering the connectivities between entities which are unrelated to the item recommendation. In addition, we also plan to adopt more datasets to investigate the scalability of our proposal in various application scenarios.

Author Contributions: Conceptualization, Z.P.; methodology, Z.P.; validation, Z.P.; writing—original draft, Z.P.; investigation, H.C.; writing—reviewing and editing, H.C.; supervision, H.C.; resources, H.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Postgraduate Scientific Research Innovation Project of Hunan Province under No. CX20200055.

Data Availability Statement: The datasets can be found at http://jmcauley.ucsd.edu/data/amazon/ (accessed on 20 June 2021) and http://www.cp.jku.at/datasets/LFM-1b/ (accessed on 20 June 2021).

Acknowledgments: The authors thank the editor and the anonymous reviewers for their valuable suggestions that have significantly improved this study.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhang, S.; Yao, L.; Sun, A.; Tay, Y. Deep Learning Based Recommender System: A Survey and New Perspectives. *ACM Comput. Surv.* **2019**, *52*, 1–38. [CrossRef]
2. Bhaskaran, S.; Marappan, R.; Santhi, B. Design and Analysis of a Cluster-Based Intelligent Hybrid Recommendation System for E-Learning Applications. *Mathematics* **2021**, *9*, 197. [CrossRef]
3. Xu, Y.; Ren, J.; Zhang, Y.; Zhang, C.; Shen, B.; Zhang, Y. Blockchain Empowered Arbitrable Data Auditing Scheme for Network Storage as a Service. *IEEE Trans. Serv. Comput.* **2020**, *13*, 289–300. [CrossRef]
4. Wang, S.; Hu, L.; Wang, Y.; Cao, L.; Sheng, Q.Z.; Orgun, M.A. Sequential Recommender Systems: Challenges, Progress and Prospects. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI '19), Macao, China, 10–16 August 2019; pp. 6332–6338.
5. Wang, B.; Cai, W. Attention-Enhanced Graph Neural Networks for Session-Based Recommendation. *Mathematics* **2020**, *8*, 1607. [CrossRef]
6. He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; Chua, T. Neural Collaborative Filtering. In Proceedings of the International World Wide Web Conference (WWW '17), Perth, Australia, 3–7 April 2017; ACM: New York, NY, USA, 2017; pp. 173–182.
7. Wang, X.; He, X.; Wang, M.; Feng, F.; Chua, T. Neural Graph Collaborative Filtering. In Proceedings of the International ACM/SIGIR Conference on Research and Development in Information Retrieval (SIGIR '19), Paris, France, 21–25 July 2019; ACM: New York, NY, USA, 2019; pp. 165–174.
8. Xu, Y.; Zeng, Q.; Wang, G.; Zhang, C.; Ren, J.; Zhang, Y. An efficient privacy-enhanced attribute-based access control mechanism. *Concurr. Comput. Pract. Exp.* **2020**, *32*, e5556. [CrossRef]
9. Wang, H.; Zhang, F.; Wang, J.; Zhao, M.; Li, W.; Xie, X.; Guo, M. RippleNet: Propagating User Preferences on the Knowledge Graph for Recommender Systems. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM '18), Torino, Italy, 22–26 October 2018; ACM: New York, NY, USA, 2018; pp. 417–426.
10. Wang, X.; He, X.; Cao, Y.; Liu, M.; Chua, T. KGAT: Knowledge Graph Attention Network for Recommendation. In Proceedings of the International Conference on Knowledge Discovery & Data Mining (KDD '19), Anchorage, AK, USA, 4–8 August 2019; ACM: New York, NY, USA, 2019; pp. 950–958.
11. Ma, X.; Dong, L.; Wang, Y.; Li, Y.; Sun, M. AIRC: Attentive Implicit Relation Recommendation Incorporating Content Information for Bipartite Graphs. *Mathematics* **2020**, *8*, 2132. [CrossRef]
12. Zhang, F.; Yuan, N.J.; Lian, D.; Xie, X.; Ma, W. Collaborative Knowledge Base Embedding for Recommender Systems. In Proceedings of the International Conference on Knowledge Discovery & Data Mining (KDD '16), San Francisco, CA, USA, 13–17 August 2016; ACM: New York, NY, USA, 2016; pp. 353–362.
13. Zhang, Y.; Ai, Q.; Chen, X.; Wang, P. Learning over Knowledge-Base Embeddings for Recommendation. *arXiv* **2018**, arXiv:1803.06540.
14. Cao, Y.; Wang, X.; He, X.; Hu, Z.; Chua, T. Unifying Knowledge Graph Learning and Recommendation: Towards a Better Understanding of User Preferences. In Proceedings of the World Wide Web Conference (WWW '19), San Francisco, CA, USA, 13–17 May 2019; ACM: New York, NY, USA, 2019; pp. 151–161.

15. Wang, H.; Zhang, F.; Zhao, M.; Li, W.; Xie, X.; Guo, M. Multi-Task Feature Learning for Knowledge Graph Enhanced Recommendation. In Proceedings of the World Wide Web Conference (WWW '19), San Francisco, CA, USA, 13–17 May 2019; ACM: New York, NY, USA, 2019; pp. 2000–2010.
16. Hu, B.; Shi, C.; Zhao, W.X.; Yu, P.S. Leveraging Meta-path based Context for Top- N Recommendation with A Neural Co-Attention Model. In Proceedings of the International Conference on Knowledge Discovery & Data Mining (KDD '18), London, UK, 19–23 August 2018; ACM: New York, NY, USA, 2018; pp. 1531–1540.
17. Sun, Z.; Yang, J.; Zhang, J.; Bozzon, A.; Huang, L.; Xu, C. Recurrent knowledge graph embedding for effective recommendation. In Proceedings of the ACM Conference on Recommender Systems (RecSys '18), Vancouver, BC, Canada, 2–7 October 2018; ACM: New York, NY, USA, 2018; pp. 297–305.
18. Wang, X.; Wang, D.; Xu, C.; He, X.; Cao, Y.; Chua, T. Explainable Reasoning over Knowledge Graphs for Recommendation. In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI '19), Honolulu, HI, USA, 27 January–1 February 2019; pp. 5329–5336.
19. Yu, X.; Ren, X.; Sun, Y.; Gu, Q.; Sturt, B.; Khandelwal, U.; Norick, B.; Han, J. Personalized entity recommendation: A heterogeneous information network approach. In Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM '14), New York, NY, USA, 24–28 February 2014; ACM: New York, NY, USA, 2014; pp. 283–292.
20. Zhao, H.; Yao, Q.; Li, J.; Song, Y.; Lee, D.L. Meta-Graph Based Recommendation Fusion over Heterogeneous Information Networks. In Proceedings of the International Conference on Knowledge Discovery & Data Mining (KDD '17), Halifax, NS, Canada, 13–17 August 2017; ACM: New York, NY, USA, 2017; pp. 635–644.
21. Wang, H.; Zhao, M.; Xie, X.; Li, W.; Guo, M. Knowledge Graph Convolutional Networks for Recommender Systems. In Proceedings of the World Wide Web Conference (WWW '19), San Francisco, CA, USA, 13–17 May 2019; ACM: New York, NY, USA, 2019; pp. 3307–3313.
22. Wang, H.; Zhang, F.; Zhang, M.; Leskovec, J.; Zhao, M.; Li, W.; Wang, Z. Knowledge-aware Graph Neural Networks with Label Smoothness Regularization for Recommender Systems. In Proceedings of the International Conference on Knowledge Discovery & Data Mining (KDD '19), Anchorage, AK, USA, 4–8 August 2019; ACM: New York, NY, USA, 2019; pp. 968–977.
23. van den Oord, A.; Li, Y.; Vinyals, O. Representation Learning with Contrastive Predictive Coding. *arXiv* **2018**, arXiv:1807.03748.
24. Devlin, J.; Chang, M.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL '19), Minneapolis, MN, USA, 2–7 June 2019; pp. 4171–4186.
25. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed Representations of Words and Phrases and their Compositionality. In Proceedings of the Conference on Neural Information Processing Systems (NeurIPS '13), Lake Tahoe, NV, USA, 5–8 December 2013; pp. 3111–3119.
26. van den Oord, A.; Kalchbrenner, N.; Espeholt, L.; Kavukcuoglu, K.; Vinyals, O.; Graves, A. Conditional Image Generation with PixelCNN Decoders. In Proceedings of the Conference on Neural Information Processing Systems (NeurIPS '16), Barcelona, Spain, 5–8 December 2016; pp. 4790–4798.
27. Chen, T.; Kornblith, S.; Norouzi, M.; Hinton, G.E. A Simple Framework for Contrastive Learning of Visual Representations. In Proceedings of the 37th International Conference on Machine Learning (ICML '20), Virtual Event, 13–18 July 2020; pp. 1597–1607.
28. Logeswaran, L.; Lee, H. An efficient framework for learning sentence representations. In Proceedings of the International Conference on Learning Representations (ICLR '18), Vancouver, BC, Canada, 30 April–3 May 2018.
29. He, K.; Fan, H.; Wu, Y.; Xie, S.; Girshick, R.B. Momentum Contrast for Unsupervised Visual Representation Learning. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR '20), Seattle, WA, USA, 13–19 June 2020; pp. 9726–9735.
30. Kingma, D.P.; Welling, M. Auto-Encoding Variational Bayes. In Proceedings of the International Conference on Learning Representations (ICLR '14), Banff, AB, Canada, 14–16 April 2014.
31. Liang, D.; Krishnan, R.G.; Hoffman, M.D.; Jebara, T. Variational Autoencoders for Collaborative Filtering. In Proceedings of the International World Wide Web Conference (WWW '18), Lyon, France, 23–27 April 2018; ACM: New York, NY, USA, 2018; pp. 689–698.
32. Goodfellow, I.J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.C.; Bengio, Y. Generative adversarial networks. *Commun. ACM* **2020**, *63*, 139–144. [CrossRef]
33. Gutmann, M.; Hyvärinen, A. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS '10), Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010; pp. 297–304.
34. Hjelm, R.D.; Fedorov, A.; Lavoie-Marchildon, S.; Grewal, K.; Bachman, P.; Trischler, A.; Bengio, Y. Learning deep representations by mutual information estimation and maximization. In Proceedings of the International Conference on Learning Representations (ICLR '19), New Orleans, LA, USA, 6–9 May 2019.
35. Yao, T.; Yi, X.; Cheng, D.Z.; Yu, F.; Chen, T.; Menon, A.; Hong, L.; Chi, E.H.; Tjoa, S.; Kang, J.; et al. Self-supervised Learning for Large-scale Item Recommendations. *arXiv* **2020**, arXiv:2007.12865.
36. Wu, J.; Wang, X.; Feng, F.; He, X.; Chen, L.; Lian, J.; Xie, X. Self-supervised Graph Learning for Recommendation. *arXiv* **2020**, arXiv:2010.10783.

37. Yu, J.; Yin, H.; Li, J.; Wang, Q.; Hung, N.Q.V.; Zhang, X. Self-Supervised Multi-Channel Hypergraph Convolutional Network for Social Recommendation. In Proceedings of the World Wide Web Conference 2021 (WWW '21), Ljubljana, Slovenia, 19–23 April 2021; ACM: New York, NY, USA, 2021; pp. 413–424.
38. Xia, X.; Yin, H.; Yu, J.; Wang, Q.; Cui, L.; Zhang, X. Self-Supervised Hypergraph Convolutional Networks for Session-based Recommendation. In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI '21), Virtual Event, 2–9 February 2021; pp. 4503–4511.
39. Zhou, K.; Wang, H.; Zhao, W.X.; Zhu, Y.; Wang, S.; Zhang, F.; Wang, Z.; Wen, J. S3-Rec: Self-Supervised Learning for Sequential Recommendation with Mutual Information Maximization. In Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), Virtual Event, 19–23 October 2020; ACM: New York, NY, USA, 2020; pp. 1893–1902.
40. Ma, J.; Zhou, C.; Yang, H.; Cui, P.; Wang, X.; Zhu, W. Disentangled Self-Supervision in Sequential Recommenders. In Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD '20), Virtual Event, 23–27 August 2020; ACM: New York, NY, USA, 2020; pp. 483–491.
41. He, X.; Deng, K.; Wang, X.; Li, Y.; Zhang, Y.; Wang, M. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In Proceedings of the International ACM/SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), Virtual Event, 25–30 July 2020; ACM: New York, NY, USA, 2020; pp. 639–648.
42. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536. [CrossRef]
43. He, R.; McAuley, J.J. VBPR: Visual Bayesian Personalized Ranking from Implicit Feedback. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI '16), Phoenix, AZ, USA, 12–17 February 2016; pp. 144–150.
44. Bollacker, K.D.; Cook, R.P.; Tufts, P. Freebase: A Shared Database of Structured General Human Knowledge. In Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence (AAAI '07), Vancouver, BC, Canada, 22–26 July 2007; pp. 1962–1963.
45. Bollacker, K.D.; Evans, C.; Paritosh, P.; Sturge, T.; Taylor, J. Freebase: A collaboratively created graph database for structuring human knowledge. In Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD '08), Vancouver, BC, Canada, 10–12 June 2008; ACM: New York, NY, USA, 2008; pp. 1247–1250.
46. Rendle, S.; Freudenthaler, C.; Gantner, Z.; Schmidt-Thieme, L. BPR: Bayesian Personalized Ranking from Implicit Feedback. In Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI '09), Montreal, QC, Canada, 18–21 June 2009; pp. 452–461.
47. Wang, Z.; Lin, G.; Tan, H.; Chen, Q.; Liu, X. CKAN: Collaborative Knowledge-aware Attentive Network for Recommender Systems. In Proceedings of the International ACM/SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), Virtual Event, 25–30 July 2020; ACM: New York, NY, USA, 2020; pp. 219–228.
48. Lin, Y.; Liu, Z.; Sun, M.; Liu, Y.; Zhu, X. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI'15), Austin, TX, USA, 25–30 January 2015; pp. 2181–2187.
49. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the International Conference on Learning Representations (ICLR '15), San Diego, CA, USA, 7–9 May 2015.
50. Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS '10), Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010; pp. 249–256.
51. Wang, F.; Xiang, X.; Cheng, J.; Yuille, A.L. NormFace: L2 Hypersphere Embedding for Face Verification. In Proceedings of the International Conference on Multimedia (MM '17), Mountain View, CA, USA, 23–27 October 2017; ACM: New York, NY, USA, 2017; pp. 1041–1049.