Knowledge-Aware Hypergraph Neural Network for Recommender Systems

Binghao Liu[1], Pengpeng Zhao[1(✉)], Fuzhen Zhuang[2,3], Xuefeng Xian[4], Yanchi Liu[5], and Victor S. Sheng[6]

[1] Institute of Artificial Intelligence, School of Computer Science and Technology, Soochow University, Suzhou, China
bhliu1@stu.suda.edu.cn, ppzhao@suda.edu.cn
[2] Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS), Institute of Computing Technology, CAS, Beijing 100190, China
[3] Xiamen Data Intelligence Academy of ICT, CAS, Beijing, China
zhuangfuzhen@ict.ac.cn
[4] Suzhou Vocational University, Suzhou, China
xianxuefeng@jssvc.edu.cn
[5] Rutgers University, New Brunswick, NJ, USA
yanchi.liu@rutgers.edu
[6] Department of Computer Science, Texas Tech University, Lubbock, USA
Victor.Sheng@ttu.edu

Abstract. Knowledge graph (KG) has been widely studied and employed as auxiliary information to alleviate the cold start and sparsity problems of collaborative filtering in recommender systems. However, most of the existing KG-based recommendation models suffer from the following drawbacks, i.e., insufficient modeling of high-order correlations among users, items, and entities, and simple aggregation strategies which fail to preserve the relational information in the neighborhood. In this paper, we propose a Knowledge-aware Hypergraph Neural Network (KHNN) framework to tackle the above issues. First, the knowledge-aware hypergraph structure, which is composed of hyperedges, is employed for modeling users, items, and entities in the knowledge graph with explicit hybrid high-order correlations. Second, we propose a novel knowledge-aware hypergraph convolution method to aggregate different knowledge-based neighbors in hyperedge efficiently. Moreover, it can conduct the embedding propagation of high-order correlations explicitly and efficiently in knowledge-aware hypergraph. Finally, we apply the proposed model on three real-world datasets, and the empirical results demonstrate that KHNN can achieve the best improvements against other state-of-the-art methods.

Keywords: Recommender systems · Knowledge-aware hypergraph · Knowledge graph

C. S. Jensen et al. (Eds.): DASFAA 2021, LNCS 12683, pp. 132–147, 2021.
https://doi.org/10.1007/978-3-030-73200-4_9

1 Introduction

With the rapid development of the Internet, recommender systems (RS) [5,10,19,23] have been widely deployed to alleviate the impact of information overloading [13]. A traditional recommendation method is collaborative filtering (CF), in which users and items are represented as ID-based vectors, and then the historical interactions of users and items are modeled by a operation such as inner product [15] or a network such as neural collaborative filtering [5]. However, the cold-start and sparsity problems generally exist in CF-based models. To address these issues, multiple types of side information have been explored for improving recommendation performance, such as item attributes [14], item reviews [26], and users' social networks [11].

Knowledge graph (KG), which is strong to model comprehensive side information, has attracted more and more attention in RS [16,19,20,22]. How to efficiently integrate the side information into latent representation vectors of users and items is important in the combination of knowledge graph and recommender systems. According to the methods dealing with the issue, existing KG-based recommender system models can be categorized into two types, path-based and graph neural network (GNN) based models. Path-based models [21] explore multiple meta-paths between target user and item in KG to infer user preference. Nevertheless, these models require domain knowledge. What's more, this type of models neglect abundant structural information stored in KG and cannot well explore and utilize the comprehensive correlations between the target user and item.

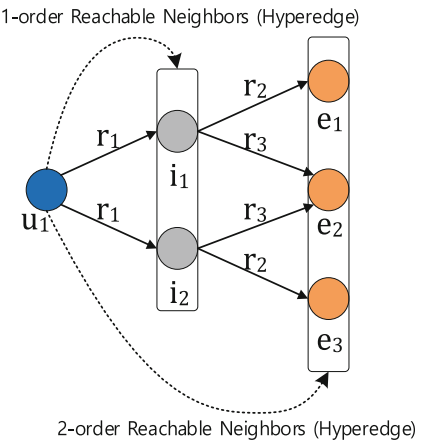


Fig. 1. An example of knowledge-aware hypergraph of u_1. u_1 is the target user whom we need to provide recommendation for. i_1 and i_2 are the items, and e_1, e_2, and e_3 are the entities in the knowledge graph. The link r_1 means i_1 and i_2 are interacted by u_1, r_2 and r_3 are defined relations in the knowledge graph.

As more and more graph neural networks emerge, several GNN-based models [20,22] have been proposed and indicate satisfactory improvements by model explicit high-order connectivity among entities in KG. However, these models still have two restrictions: (**L1**) High-order correlations between users, items, and entities in KG are essential for data modeling. These methods mainly apply GNN to enrich the representation of a target node by recursively aggregating their nearest neighbors in the KG, so have limitations on using high-order correlation between the target node and unoriginal neighbors. In addition, the graph structure, which is employed by existing methods, are insufficient to model and utilize high-order relation, as a typical graph can only store pairwise connections. (**L2**) During the neighborhood aggregation, simple aggregation operations such as element-wise mean/sum or max-pooling or weighted aggregation operations which only consider pair-wise connection are usually used as the aggregator to get the neighborhood embedding. This aggregation destroys rich relational information among neighborhood nodes. As a result, The neighborhood aggregation is forced to ignore the fine structure of neighborhood relations.

To address the above limitation of existing KG-based recommendation methods, we propose an end-to-end model named Knowledge-aware Hypergraph Neural Network (KHNN). Specifically, KHNN is equipped with two designs: (1) Knowledge-aware hypergraph construction, which is composed of initial hyperedge construction and knowledge hyperedge construction. For initial hyperedge convolution, we construct hyperedges based on the user-item interaction. For knowledge hyperedge construction, we construct hyperedges based on knowledge graph and initial hyperedge. Based on these generated hyperedges, we can construct two hypergraphs for the target user and item, respectively. As depicted in Fig. 1, item i_1 and item i_2 can be associated by target user u_1 without direct connections, and entity e_1, entity e_2, and entity e_3 in knowledge graph can be associated by item i_1 and item i_2 without direct connections. (**L1**) (2) Knowledge-aware hypergraph convolution, which is composed of neighborhood convolution and hyperedge convolution. For neighborhood convolution, we use a transform matrix to permute and weight entities in a hyperedge. For hyperedge convolution, we just concatenate the hyperedge features to get the final embeddings of users and items. (**L2**) To sum up, our contributions are as follows:

- We propose a knowledge-aware hypergraph which explicitly exploring and modeling the high-order correlations among users, items, and entities in the KG.
- We propose to apply a novel knowledge-aware hypergraph convolution method to model the rich relational information among neighborhood entities in a hyperedge and support the explicit and efficient embedding propagation of high-order correlations.
- We conduct sufficient experiments on all three recommendation scenarios. The results of the experiments show the superiority of KHNN over other state-of-the-art baselines.

2 Related Work

In this section, we will introduce the representative models in knowledge aware recommendation and hypergraph learning.

2.1 Knowledge-Aware Recommendation

Recently, Knowledge graph get more and more attention. Kg is widely applied in recommender systems for extending correlations between user historical interaction and candidate items. Some models apply KG-aware embeddings to improve the quality of item embeddings such as DKN [17] and CKE [25]. Nevertheless, these methods are insufficient to make use of high-order relations over KG. As a result, several models [21,24] have been applied to explore multiple semantic path (meta-path), which connect target users' historical interactions and candidate items over KG. Then the prediction function is learned through multiple path modeling and integrating. In spite of these models' effectiveness, the path-based models neglects comprehensive relational information stored in KG as each explored path is modeled independently. As a newly research domain, graph neural network has demonstrated its ability in learning good node embeddings in graph. RippleNet [16] diffuses users' underlying interest and explores their rich correlations in KG. KGCN-LS [18] and KGAT [20] conduct high-order information propagation by applying multiple KG-aware GNN layers. CKAN [22] utilizes the collaborative information by collaborative information propagation through users' historical interactions and it also apply a solution to combine collaborative information with knowledge information. Though GNN based models have obtained some performance improvement, they still suffer inferior performance, i.e., insufficient modeling of high-order correlations among users, items and entities in knowledge graph and simple aggregation strategies which fail to preserve the relational information in the neighborhood.

2.2 Hypergraph Learning

As a label propagation method, [27] first propose hypergraph learning to complex high-order relations. With the development of deep learning, hypergraph neural network have received much attention. Recent works focus on the learning of hyperedge weight. In these methods, the hyperedges or sub-hypergraphs with higher significance will be allocated larger weight [3]. Hypergraph neural network (HGNN) has been proposed as the first method to apply graph convolution on hypergraph structure [2]. Besides learning label propagation on hypergraph, dynamic hypergraph neural networks (DHGNN) [7] has been proposed to explore and model high-order relations among vertex in hypergraph. In addition, hypergraph has been applied in recommendation. In dual channel hypergraph collaborative filtering (DHCF) [6], hypergraph is used to learn the embeddings of users and items, as a result, these two types of information can be well connected while still maintaining their own features. Inspired by [6] and [7], we propose a way to apply the Knowledge-aware hypergraph for explicitly constructing

knowledge-aware hyperedges to utilize the high-order correlations among users, items, and entities in KG.

3 Problem Definition

In this section, we introduce the formulation of the Knowledge-aware Hypergraph Neural Network. In the Knowledge-aware recommendation scenario, M users are represented as $\mathcal{U} = \{u_1, u_2, ..., u_M\}$, N items are represented as $\mathcal{V} = \{v_1, v_2, ..., v_N\}$ and a user-item interaction are represented as matrix $\mathrm{Y} \in \mathbb{R}^{M \times N}$. The matrix Y is constructed according to users' implicit feedback, in which $y_{uv} = 1$ indicates that user u has interacted with item v, such as clicking, collecting, or purchasing; otherwise $y_{uv} = 0$. In addition, there is also a knowledge graph $\mathcal{G}_K = \{(h, r, t)|h, t \in \mathcal{E}, r \in \mathcal{R}\}$. In the knowledge graph, each knowledge triple (h, r, t) demonstrates that a relationship r exist between head entity h and tail entity t. And the sets of entities and relations in the knowledge graph are denoted as $\mathcal{E}$ and $\mathcal{R}$. For instance, the triple (*Back to the Future, directedby, Robert Zemeckis*) represents the fact that *Robert Zemeckis* is an director of the movie *Back to the Future*.

An edge can only connects two nodes in a typical graph. However, a hyperedge connects two or more nodes in a hypergraph [1]. $\mathcal{G}_H = (\mathcal{E}, \mathcal{H})$ denotes a knowledge-aware hypergraph, in which $\mathcal{E}$ denotes the entity set, and $\mathcal{H}$ represents the hyperedge set. A set $\mathcal{A} = \{(v, e)|v \in \mathcal{V}, e \in \mathcal{E}\}$ is used to conduct correlations between items and entities. In A, (v, e) indicates that item v can be correlated with entity e in the knowledge-aware hypergraph.

Given the knowledge-aware hypergraph $\mathcal{G}_H$ with historical interaction matrix Y, we aim to calculate the probability that user u would interact item v which he has not engaged with before. To be specific, our ultimate goal is to learn a prediction function $\hat{y}_{uv} = \mathcal{F}(u, v|\Theta, Y, \mathcal{G}_H)$, where $\hat{y}_{uv}$ denotes the predicted probability that user u will interact item v, and Θ indicates the model parameters of function $\mathcal{F}$.

4 The Proposed Method

In this section, the proposed KHNN framework is introduced. As represented in Fig. 2, the model contains three main components: 1) knowledge-aware hypergraph construction, which consists of initial hyperedge construction through user-item interactions and knowledge hyperedge construction in knowledge graph; 2) knowledge-aware hypergraph convolution, which consists of neighborhood convolution which aggregates entities to hyperedge and hyperedge convolution which aggregates vectors of hyperedge to user u and item v; and 3) prediction layer, which outputs the predicted click probability.

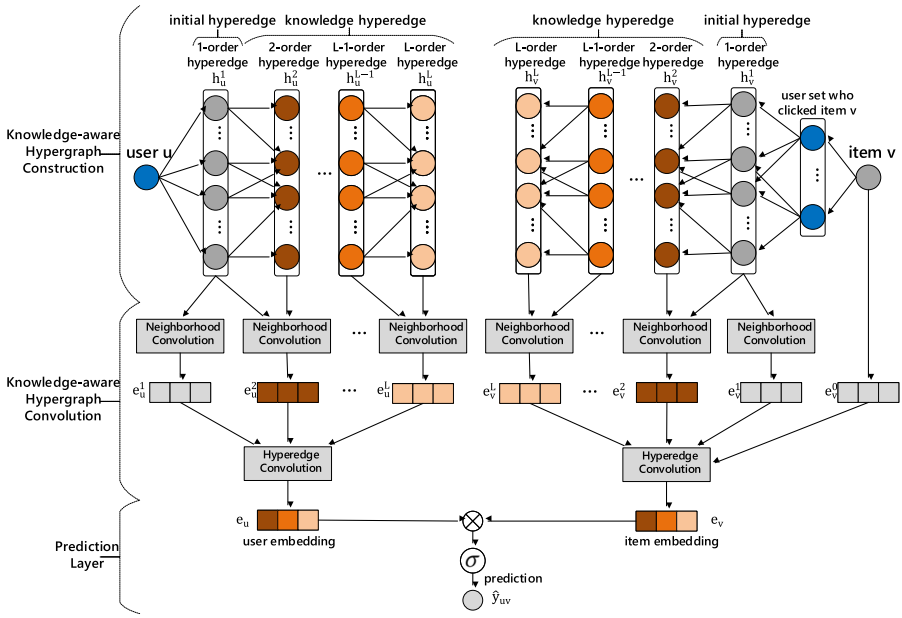


Fig. 2. The proposed KHNN model consists of three components: knowledge-aware hypergraph construction, knowledge-aware hypergraph convolution and prediction layer.

4.1 Knowledge-Aware Hypergraph Construction

As shown in Fig. 2, knowledge-aware hypergraph construction is composed of two major modules: initial hyperedge construction and knowledge hyperedge construction. The initial hyperedge is constructed by explicitly encoding user-item historical interaction into representations of both user and item. The knowledge hyperedge is constructed by entities which have high-order relation with source node in knowledge-aware hypergraph.

Initial Hyperedge Construction. It is obvious that users' historical interactions are able to representing the user's interest to some extent. As a result, we use user's interacted items to represent user u. The interacted item set of user u can be constructed as the initial hyperedge(entity set) in knowledge-aware hypergraph through the correlations between items and entities. Following, the initial hyperedge of user u is defined as:

$$h_u^1 = \{e|(v,e) \in \mathcal{A} \; and \; v \in \{v|y_{uv} = 1\}\} \tag{1}$$

Simplistically, users who have similar historical interactions can also be used to enrich the item's feature representation. We use items, which have been watched by the same user, to construct initial item set of item v as follows:

$$\mathcal{V}_v = \{v_u|u \in \{u|y_{uv} = 1\} \, and \; y_{uv_u} = 1\} \tag{2}$$

Integrating initial item set and correlation set, the initial hyperedge of item v is defined as follows:

$$h_v^1 = \{e|(v_u, e) \in \mathcal{A} \; and \; v_u \in \mathcal{V}_v\} \tag{3}$$

The user and item's initial hyperedges are constructed now for the following knowledge hyperedge construction.

Knowledge Hyperedge Construction. Entities in KG have close relations with its neighborhood. We construct the knowledge hyperedge by taking advantage of such close relations among entities in KG. As a result, knowledge hyperedges of different distances from the initial hyperedge are able to enrich the latent embedding of user and item efficiently. The knowledge hyperedge for user u and item v is recursively defined as:

$$h_o^l = \left\{t|(h, r, t) \in \mathcal{G}_K \; and \; h \in h_o^{l-1}\right\}, \quad l = 2, 3, ..., L \tag{4}$$

where l indicates the distance from the initial hyperedge, the subscript symbol o[1] is a uniform placeholder for symbol u or v.

Applying knowledge graph as auxiliary information to construct knowledge hyperedges is helpful to enrich the features of user and item. As illustrated in Fig. 2, the hyperedge is obtained through initial hyperedge construction and knowledge hyperedge construction. Based on these generated hyperedges, we can construct two hypergraphs for target user and target item.

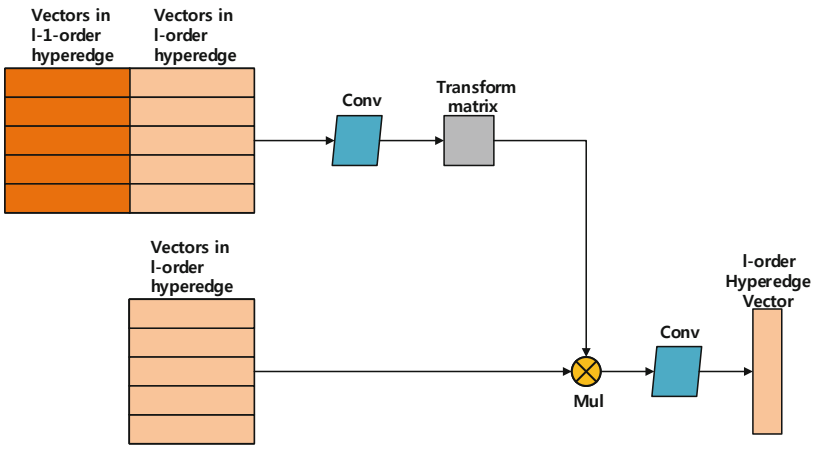


Fig. 3. Neighborhood convolution module. For k concat vectors, a k × k transform matrix is computed by convolution. We multiply transform matrix and input vector matrix to get permuted and weighted vector matrix. Then we apply a 1-dimension convolution to weighted vector matrix to get the 1-dimension hyperedge vector.

[1] Symbol o is used as a uniform placeholder for both user and item.

4.2 Knowledge-Aware Hypergraph Convolution

Knowledge-aware hypergraph convolution is composed of two sub-modules: neighborhood convolution sub-module and hyperedge convolution sub-module. Neighborhood convolution aggregates entity latent vectors to hyperedge and then hyperedge convolution aggregates hyperedge latent vectors to a single embedding for both user and item.

Neighborhood Convolution. Neighborhood convolution aggregates entity latent vectors to the hyperedge. A simple solution is pooling method like max pooling and average pooling. State-of-the-art methods like CKAN [22] employ a fixed, pre-computed transform matrix generated from triples in kg for vertex aggregation. However, such methods are unable to model comprehensive correlation among entities in neighborhood well. Inspired by DHGNN [7], we propose a novel neighborhood convolution method. We learn the transform matrix T from the entity vectors in both l-order and l-1-order hyperedges for vector permutation and weighting, as is shown in Fig. 3. The transform matrix obtained by convolution can make good use of the comprehensive correlation among entities in neighborhood. To be specific, we use a 1-d convolution to generate transform matrix T and use another 1-d convolution to aggregate the transformed vectors, as is described by Eq. 5 and Eq. 6.

$$T_o^l = conv_1(h_o^{l-1}||h_o^l) \quad l = 2, 3, ..., L \tag{5}$$

$$e_o^l = conv_2(T_o^l \cdot h_o^l) \quad l = 2, 3, ..., L \tag{6}$$

where the subscript o is a uniform placeholder for both user and item. $||$ is the concatenation operation. $conv_1$ and $conv_2$ are 1-dimension convolution but with different out channels.

Notice that, because the initial hyperedge has strong connections with original user and item. Consequently, the initial hyperedge are added for both user and item, as is described by Eq. 7 and Eq. 8.

$$T_o^1 = conv_1(h_o^1) \tag{7}$$

$$e_o^1 = conv_2(T_o^1 \cdot h_o^1) \tag{8}$$

It is noted that item v has its associated entities in kg to represent item v itself while user u does not. The entity, which have direct correlation with item v, contains most useful information to represent item v itself in kg. Consequently, the entity is added to the representation set of the item v and the entity is formulated as follows:

$$e_v^0 = e \quad (v, e) \in \mathcal{A} \tag{9}$$

After neighborhood convolution, we define the user representation set containing hyperedge embeddings and item representation set containing both hyperedge embeddings and additional embeddings for item v itself as follows:

$$\mathcal{T}_u = \left\{e_u^1, e_u^2, ..., e_u^L\right\} \tag{10}$$

$$\mathcal{T}_v = \left\{e_v^0, e_v^1, e_v^2, ..., e_v^L\right\} \tag{11}$$

Hyperedge Convolution. Hyperedge convolution aggregates hyperedge vector to user u and item v, as is illustrated in Fig. 2. We apply three types of aggregators to aggregate the multiple embeddings of hyperedge in Eq. 10 and Eq. 11 into a single embedding for both user and item.

Sum aggregator perform summation calculations of multiple embeddings in the set of representations:

$$agg^{o}_{sum} = \sum_{e_o \in \mathcal{T}_o} e_o \tag{12}$$

Pooling aggregator perform element-wise maximization calculations of multiple embeddings in the set of representations:

$$agg^{o}_{pool} = pool_{max}(\mathcal{T}_o) \tag{13}$$

Concat aggregator concatenates multiple embeddings in the set of representations:

$$agg^{o}_{concat} = e^{i_1}_{o} || e^{i_2}_{o} || ... || e^{i_n}_{o} \tag{14}$$

where $e^{i_k}_{o} \in \mathcal{T}_o$, and $||$ is the concatenation operation. Specifically, since the precondition of inner product operation is same dimensions of embeddings for both user and item, we abandon the initial hyperedge embedding in the item representation set to concatenate the rest representation vectors. The detail we will discuss in Sect. 5.5.

4.3 Optimization

Model Prediction. The single embedding of user's knowledge-aware hypergraph is defined as e_u. Analogously, as to item, e_v denote the item's knowledge-aware hypergraph. At last, we calculate the inner product and nonlinear transformation of embeddings to predict the user's score for the candidate item:

$$\hat{y}_{uv} = \sigma(e^{T}_{u} e_v) \tag{15}$$

Loss Function. For each user, we random select the same number of negative samples with positive samples to make sure the effectivity of model training. We will discuss the details of negative sampling in Sect. 5.1. Ultimately, we define the loss function of the model KHNN as follows:

$$\mathcal{L} = \sum_{u \in U} (\sum_{v \in \{v|(u,v) \in \mathcal{P}^+\}} \mathcal{J}(y_{uv}, \hat{y}_{uv}) - \sum_{v \in \{v|(u,v) \in \mathcal{P}^-\}} \mathcal{J}(y_{uv}, \hat{y}_{uv})) + \lambda ||\Theta||^2_2 \tag{16}$$

$\mathcal{J}$ is the cross-entropy loss, $\mathcal{P}^+$ is positive samples while $\mathcal{P}^-$ means the negative samples. Θ is the model parameters, and $||\Theta||^2_2$ is the L2-regularizer that parameterized by λ.

Table 1. Statistics of all the three datasets: Last.FM (Music), BookCrossing (Book) and Movie-Lens20M (Movie).

	Music	Book	Movie
# users	1,872	17,860	138,159
# items	3,846	14,967	16,954
# interactions	42,346	139,746	13,501,622
# inter-avg	23	8	98
# entities	9,366	77,903	102,569
# relations	60	25	32
# KG triples	15,518	15,1500	499,474
# link-avg	4	10	29

5 Experiments

In this section, we evaluate the KHNN model under three real-world scenarios to answer the following research questions:

- **Q1:** How does KHNN perform compared with state-of-the-art KG-based recommendation methods?
- **Q2:** How do different components (e.g., knowledge-aware neighborhood convolution and aggregator selection in hyperedge convolution) affect KHNN?
- **Q3:** How do different hyper-parameter settings (e.g. number of hyperedge, size of hyperedge) affect KHNN?

5.1 Dataset Description

We conduct experiments under three different scenarios: music, book and movie recommendations. The three datasets are different in size, sparsity and domain.

- **Last.FM**[2] contains 2 thousand users' listening information from Last.fm online music platform.
- **Book-Crossing**[3] contains 1 million ratings (ranging from 0 to 10) of books in the Book-Crossing community.
- **MovieLens-20M**[4] that contains 138 thousand users who have watched 27 thousand movies with 20 million ratings (ranging from 1 to 5) on the MovieLens platform.

The historical interactions in these three datasets are explicit feedback, as a result, these historical records need to be transformed into the implicit records in which 1 indicates the positive feedback. With regard to negative samples, we

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

randomly select same size with user's positive feedback from items which he has not interacted with.

The sub-KGs is constructed to guarantee the quality of entities in knowledge-aware hypergraph. We follow the work in [16,19] to construct sub-KGs from the Satori[5]. Each sub-KG, in which the confidence level of triple is higher than 0.9, is a subset of the whole KG. We abandon the items correlated with multiple entities and the items associated with no entity for accuracy. Table 1 is the summary of detailed statistics of the three datasets: Last.FM (music), Book-Crossing (book), MovieLens-20M (movie).

5.2 Baselines

We compare KHNN with following four types of recommendation models: CF-based method (BPRMF), embedding-based model (CKE), path-based model (PER) and GNN-based models(RippleNet, KGCN, KGNN-LS, KGAT, CKAN).

- **BPRMF** [12] is the Bayesian ranking model that uses Matrix Factorization (MF) as the prediction component for recommendation. It utilizes the user-item interaction to learn representations of users and items.
- **CKE** [25] is a typical embedding-based model, which use semantic embeddings derived from TransR [9] to enhance matrix factorization [12].
- **PER** [24] is a typical path-based model which use latent features derived from the meta-path in KG to denote the correlation between users and items in kg.
- **RippleNet** [16] is a GNN-based model which propagates user's interest to learn user and item embedding.
- **KGCN** [19] is a state-of-the-art GNN-based model which models the ultimate embedding of a candidate item v by aggregating the embedding of entities in the KG from neighbors of item v to item v itself.
- **KGNN-LS** [18] is another state-of-the-art GNN-based model which learn user-specific item embeddings by identifying important knowledge graph relationships for a given user
- **KGAT** [20] is another state-of-the-art GNN-based model, which considers user nodes as one type of entities in the knowledge graph and the interaction between users and items as one type of relation
- **CKAN** [22] is another state-of-the-art GNN-based model, which utilizes the collaborative information by collaborative information propagation through users' historical interactions and it also apply a solution to combine collaborative information with knowledge information.

5.3 Experimental Settings

In our experiments, these three datasets are divided into training, evaluation, and test sets with the proportion of 6:2:2. The following are the two recommendation

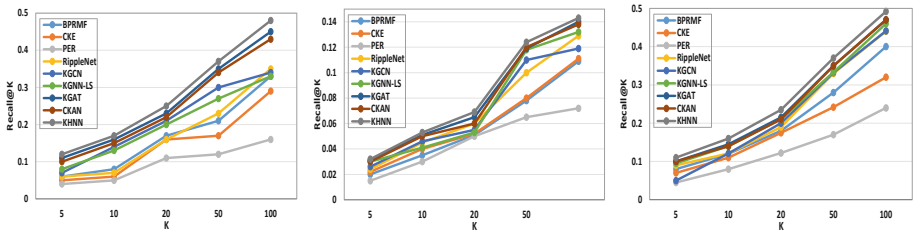
[5] https://searchengineland.com/library/bing/bing-satori.

Table 2. The result of AUC and F1 in CTR prediction.

Model	Last.FM		Book-Crossing		MovieLens-20M	
	AUC	F1	AUC	F1	AUC	F1
BPRMF	0.756	0.701	0.658	0.611	0.958	0.914
CKE	0.747	0.674	0.676	0.623	0.927	0.874
PER	0.641	0.603	0.605	0.572	0.838	0.792
RippleNet	0.776	0.702	0.721	0.647	0.976	0.927
KGCN	0.802	0.708	0.684	0.631	0.977	0.930
KGNN-LS	0.805	0.722	0.676	0.631	0.975	0.929
KGAT	0.829	0.742	0.731	0.654	0.976	0.928
CKAN	0.842	0.769	0.753	0.673	0.976	0.929
KHNN	**0.856**	**0.785**	**0.764**	**0.687**	**0.986**	**0.944**

scenarios we take into account. (1) As to click-through rate (CTR) prediction, we learn model parameters from the training set. Then we use the trained model to predict the interest of user's about the items in the test set. (2) As to top-K recommendation, the model, which learn model parameters from the training set, is used to choose K items with higher predicted score than other items in the test set. The metrics of AUC and F1 are used in CTR prediction and the metrics of Recall@K is selected in top-K recommendation. We use adam [8] to optimize all models in training. We set the batch size as 1024 in training. We apply the default Xavier initializer [4] to initialize the model's parameters.

We implement our KHNN model in PyTorch[6]. The best hyper-parameters are obtained by grid search. The learning rate is searched in $\{10^{-3}, 5\times10^{-3}, 10^{-2}, 5\times10^{-2}\}$. The embedding size is tuned among {8, 16, 32, 64, 128, 256}. The coefficient of L2 normalization is searched in $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$. We search the set size of hyperedge in {16, 32, 64, 128}. It's note that we set same size of hyperedge for user and item.



(a) Last.FM (b) Book-Crossing (c) MovieLens-20M

Fig. 4. The result of Recall@K in top-K recommendation.

[6] https://pytorch.org.

5.4 Performance Comparison (Q1)

The results of CTR prediction and top-K recommendation are presented in Table 2 and Fig. 4, respectively. Analyzing the experimental results, we can draw the following conclusions:

- KHNN shows overwhelming superiority over these state-of-the-art baselines on all three datasets. Specifically, KHNN achieve improvements over the state-of-the-art baselines w.r.t. AUC by 1.7% in Last.FM, 1.5% in Book-Crossing and 1% in MovieLens-20M.
- In CTR prediction and top-K recommendation, the KHNN's performance is excellent. Compared with CKAN, the performance of KHNN demonstrate the effectiveness of the knowledge-aware hypergraph convolution. The results' difference among KHNN, KGCN and KGNN-LS demonstrate the importance of explicitly encoding high-order correlation in knowledge graph.
- It is obvious that all models obtain the highest scores in the experiments on the MovieLens-20M dataset. One possible reason is that the number of average interactions and average links in kg are highest on the MovieLens-20M dataset. Consequently, there is not sufficient interactions and links in the poorer datasets for recommender models to learn the latent vectors specifically in kg-based models.
- According to the results of experiments, KG-based models achieve better performance than CF-based models in most cases. This experimental results illustrate that the usage of KG is helpful to capture underlying historical interaction between users and items.
- According to the results of KG-based models, the GNN-based models achieve better performance than the path-based models on all three datasets. This demonstrates the importance of modeling the high-order connectivity of neighbors in knowledge graph.

5.5 Study of KHNN (Q2)

Effect of Knowledge-Aware Neighborhood Convolution. To verify the impact of knowledge-aware neighborhood convolution, we study two variants of KHNN. For KHNN$_{mean-1}$, we apply mean aggregator to aggregate embeddings in 1-order neighborhood. For KHNN$_{mean-all}$, we apply mean aggregator to aggregate embeddings in all neighborhood. In the analyses of data from Table 3. From the results of experiments, we can observe that KHNN consistently perform better than KHNN$_{mean-1}$ and KHNN$_{mean-all}$. We attribute the improvement to the neighborhood convolution, which is able to model comprehensive correlation among entities in neighborhood well.

Effect of Aggregator Selection in Hyperedge Convolution. To explore the impact of aggregator selection in hyperedge convolution, we apply Sum, Pool and Concat aggregator in KHNN to conduct hyperedge convolution. The results of experiment are shown in Table 4. We can draw the following conclusion:

agg_{concat} is superior to agg_{sum} and agg_{pool}. The results can be owed to that the Concat aggregator is able to retain more information stored in embeddings than Sum and Pool aggregators.

Table 3. The result of AUC w.r.t. effect of knowledge-aware neighborhood convolution.

Category	KHNN$_{mean-all}$	KHNN$_{mean-1}$	agg_{concat}
Music	0.811	0.840	**0.856**
Book	0.738	0.751	**0.764**
Movie	0.964	0.970	**0.986**

Table 4. The result of AUC w.r.t. different aggregators in hyperedge convolution.

Aggregator	agg_{sum}	agg_{pool}	agg_{concat}
Music	0.835	0.844	**0.856**
Book	0.740	0.752	**0.764**
Movie	0.970	0.972	**0.986**

5.6 Hyper-Parameter Study (Q3)

Effect of Number of Hyperedge. To explore how the number of hyperedge affects the performance, We vary the maximal number of hyperedge L of hyperedge construction. What's more, the concat aggregator should use the same number of embeddings used for calculation. However, the number of embeddings in $\mathcal{T}_u$ and $\mathcal{T}_v$ is different. As a result, we exclude e_v^1 in $\mathcal{T}_v$ as a compromise solution. For example, L = 2 means to aggregate e_v^0 and e_v^2 for item hyperedge convolution. Table 5 show the results of experiment with different number of hyperedge. It is obvious that the model achieve best performance when L is 3 in music, 2 in book and 2 in movie. The best results can be owed to the construction and convolution of high-order hyperedge provides sufficient auxiliary knowledge information. However, not only information but also noise are bring when the number of hyperedge is large.

Effect of Size of Hyperedge. We set the size of hyperedge form 16 to 128. The results is presented in Table 6. The best performance of music and book is obtained when the size is taken as 64. For movie recommendation, the best result is obtained when the size is taken as 128. One reasonable explanation is that not only knowledge information but also noise are bring by hyperedge.

Table 5. The result of AUC w.r.t. different number of hyperedge.

Number of Hyperedge	1	2	3	4
Music	0.835	0.842	**0.856**	0.850
Book	0.748	**0.764**	0.753	0.745
Movie	0.971	**0.986**	0.983	0.977

Table 6. The result of AUC w.r.t. different sizes of hyperedge.

Size	16	32	64	128
Music	0.827	0.842	**0.856**	0.848
Book	0.743	0.755	**0.764**	0.752
Movie	0.959	0.968	0.979	**0.986**

6 Conclusion

In this paper, we proposed a Knowledge-aware Hypergraph Neural Network (KHNN) framework. First, the knowledge-aware hypergraph structure is employed for modeling users items and entities in knowledge graph with explicit hybrid high-order correlations. Second, a novel knowledge-aware hypergraph convolution method is proposed to aggregate different knowledge-based neighbors in hyperedge efficiently and support the explicit embedding propagation of high-order correlations in knowledge-aware hypergraph. Our extensive experimental results on three real-world datasets demonstrated the superiority of KHNN over other models. As to future work, we will focus on how to design a better way to aggregate the multiple representations in hyperedge to get better performance.

Acknowledgements. This research was partially supported by NSFC (No. 61876117, 61876217, 61872258, 61728205), ESP of the State Key Laboratory of Software Development Environment, and PAPD of Jiangsu Higher Education Institutions.

References

1. Benson, A.R., Gleich, D.F., Leskovec, J.: Higher-order organization of complex networks. Science **353**(6295), 163–166 (2016)
2. Feng, Y., You, H., Zhang, Z., Ji, R., Gao, Y.: Hypergraph neural networks. In: AAAI, pp. 3558–3565. AAAI Press (2019)
3. Gao, Y., Wang, M., Tao, D., Ji, R., Dai, Q.: 3-D object retrieval and recognition with hypergraph analysis. IEEE Trans. Image Process. **21**(9), 4290–4303 (2012)
4. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: AISTATS. JMLR Proceedings, vol. 9, pp. 249–256. JMLR.org (2010)
5. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.: Neural collaborative filtering. In: WWW, pp. 173–182. ACM (2017)

6. Ji, S., Feng, Y., Ji, R., Zhao, X., Tang, W., Gao, Y.: Dual channel hypergraph collaborative filtering. In: KDD, pp. 2020–2029. ACM (2020)
7. Jiang, J., Wei, Y., Feng, Y., Cao, J., Gao, Y.: Dynamic hypergraph neural networks. In: IJCAI, pp. 2635–2641. ijcai.org (2019)
8. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: ICLR (Poster) (2015)
9. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: AAAI, pp. 2181–2187. AAAI Press (2015)
10. Luo, A., et al.: Collaborative self-attention network for session-based recommendation. In: IJCAI, pp. 2591–2597. ijcai.org (2020)
11. Massa, P., Avesani, P.: Trust-aware recommender systems. In: RecSys, pp. 17–24. ACM (2007)
12. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: Bayesian personalized ranking from implicit feedback. In: UAI, pp. 452–461. AUAI Press (2009)
13. Ricci, F., Rokach, L., Shapira, B.: Introduction to recommender systems handbook. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P.B. (eds.) Recommender Systems Handbook, pp. 1–35. Springer, Boston, MA (2011). https://doi.org/10.1007/978-0-387-85820-3_1
14. Sen, S., Vig, J., Riedl, J.: Tagommenders: connecting users to items through tags. In: WWW, pp. 671–680. ACM (2009)
15. Wang, H., Wang, J., Zhao, M., Cao, J., Guo, M.: Joint topic-semantic-aware social recommendation for online voting. In: CIKM, pp. 347–356. ACM (2017)
16. Wang, H., et al.: RippleNet: propagating user preferences on the knowledge graph for recommender systems. In: CIKM, pp. 417–426. ACM (2018)
17. Wang, H., Zhang, F., Xie, X., Guo, M.: DKN: deep knowledge-aware network for news recommendation. In: WWW, pp. 1835–1844. ACM (2018)
18. Wang, H., et al.: Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. In: KDD, pp. 968–977. ACM (2019)
19. Wang, H., Zhao, M., Xie, X., Li, W., Guo, M.: Knowledge graph convolutional networks for recommender systems. In: WWW, pp. 3307–3313. ACM (2019)
20. Wang, X., He, X., Cao, Y., Liu, M., Chua, T.: KGAT: knowledge graph attention network for recommendation. In: KDD, pp. 950–958. ACM (2019)
21. Wang, X., Wang, D., Xu, C., He, X., Cao, Y., Chua, T.: Explainable reasoning over knowledge graphs for recommendation. In: AAAI, pp. 5329–5336. AAAI Press (2019)
22. Wang, Z., Lin, G., Tan, H., Chen, Q., Liu, X.: CKAN: collaborative knowledge-aware attentive network for recommender systems. In: SIGIR, pp. 219–228. ACM (2020)
23. Xu, C., et al.: Long- and short-term self-attention network for sequential recommendation. Neurocomputing **423**, 580–589 (2021)
24. Yu, X., et al.: Personalized entity recommendation: a heterogeneous information network approach. In: WSDM, pp. 283–292. ACM (2014)
25. Zhang, F., Yuan, N.J., Lian, D., Xie, X., Ma, W.: Collaborative knowledge base embedding for recommender systems. In: KDD, pp. 353–362. ACM (2016)
26. Zheng, L., Noroozi, V., Yu, P.S.: Joint deep modeling of users and items using reviews for recommendation. In: WSDM, pp. 425–434. ACM (2017)
27. Zhou, D., Huang, J., Schölkopf, B.: Learning with hypergraphs: clustering, classification, and embedding. In: NIPS, pp. 1601–1608. MIT Press (2006)