Information Sciences 540 (2020) 263–277

Contents lists available at ScienceDirect

Information Sciences

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

DKEN: Deep knowledge-enhanced network for recommender systems

Xiaobo Guo [a,*], Wenfang Lin [a], Youru Li [a,b,c], Zhongyi Liu [a], Lin Yang [a], Shuliang Zhao [d], Zhenfeng Zhu [b,c]

[a] Ant Financial Service Group, Beijing 100020, China
[b] Institute of Information Science, Beijing Jiaotong University, Beijing 100044, China
[c] Beijing Key Laboratory of Advanced Information Science and Network Technology, Beijing 100044, China
[d] College of Computer & Cyber Security, Hebei Normal University, ShiJiaZhuang 050010, China

ARTICLE INFO

Article history:
Received 12 October 2019
Received in revised form 12 June 2020
Accepted 14 June 2020
Available online 29 June 2020

Keywords:
Recommender systems
Knowledge graph embedding
Ensemble learning

ABSTRACT

Despite that existing *knowledge graphs embedding* (KGE) based methods can achieve better recommendation performance compared with deep learning based ones, such improvement is limited due to lack of capturing the shared information between user-item interaction and item-item relation encoded in *knowledge graph* (KG) by fully leveraging the implicit and explicit relationship. To address this issue, in this paper, we propose a principled *deep knowledge-enhanced network* (DKEN) framework based on deep learning and KGE to model the semantics of entities and relations encoded in the KG. In particular, the DKEN utilizes *deep neural networks* (DNN) to learn higher-order feature interactions and ensembles KGE features with DNN features into an end-to-end learning process naturally to exploit implicit interaction and explicitt semantic features. Furthermore, a *cross information sharing* (CIS) layer is designed to facilitate information sharing between items and entities, and two aggregators are developed to improve the performance of the model. Extensive experiments on several public datasets, as well as online AB tests of an industrial recommendation scenario in the Ant Financial Service Group, demonstrate that DKEN achieves remarkably better performance than several state-of-the-art baselines.

1. Introduction

With the fast growth of data volume, computation power and deep learning algorithms, *deep learning-based recommended systems* (DLRS) instead of collaborative filtering based methods have attracted more and more attention in both academia and industry [1,2]. Despite its effectiveness and universality, DLRS suffer from the inability of modeling user preferences without more side information. In fact, this kind of data sparsity problem is a big challenge in DLRS, especially in cold-start scenarios.

In general, side information likes user or item's attribute information [3–6], entity2rec [7], node2vec [8], knowledge graph [9–16], co-authorship [17], meta-prod2vec [18] and even user's social tweets [19] can be introduced to address the data sparsity and improve the performance of recommended systems. More typically, KGs can be usually considered as the source

* Corresponding author.
E-mail address: jefflittleguo.gxb@antfin.com (X. Guo).

https://doi.org/10.1016/j.ins.2020.06.041
0020-0255/

of side information. By doing so, more and more KGs are publicly available (*e.g.* Google Knowledge Graph,[1] Microsoft Bing Satori,[2] Baidu Knowledge Graph[3] and Sogou Knowledge Cube[4]) to research community in recent years. Furthermore, inspired by the popularity of KGs in a wide variety of tasks, existing studies have usually tried to learn the representation of entities and relations encoded in KG by KGE such as RKGE [13], KTUP [20], KGCN [21], DKN [22], RippleNet [23], MKR [24] and so on. However, a major problem with this kind of methods is how to fully exploit the semantics of relations encoded in KG to capture the explicit and implicit relationship between items and entities during the representation learning.

Based on the above, some aspects which are still yet to be explored can be summarized as follows,

- How to incorporate KGE into the end-to-end learning process of a recommendation model? Existing methods, like DKN, usually train entity embeddings separately, which can cause sub-optimal results, as KGE is not optimized towards specific tasks.
- How to capture the shared information between user-item interaction and item-item relation encoded in KG by fully exploiting explicit and implicit relationship for effective recommendation? From the perspective of item representation learning, the DLRS part models implicit semantics of items with user-item interactions while the KG part models explicit semantics of items with their relations with other entities. Since implicit and explicit semantics are highly complementary, it would be beneficial if we establish an information sharing mechanism between the two parts.
- How to design the KGE method to better represent the KG in a low-dimensional space so as to be better adapted to the recommendation task? Instead of taking a simple random sampler during the performance propagation phase, it is necessary to introduce an aggregator to obtain the user's principal preference signal in a more effective way.

To tackle the above issues, the propose DKEN, an end-to-end knowledge enhanced DLRS framework. A *cross-information share* (CIS) layer is designed to integrate KGE into DLRS and facilitate information sharing between items and entities in the embedding phase. A *knowledge-enhanced network* (KEN) layer is designed to improve the knowledge graph embedding learning of the traditional propaganda-based method [23], which features two improved sampling strategies during performance propagation operations. A *deep ripple network* (DRN) layer is designed to combine DNN [25] features and KGE features into an end-to-end model.

The key contributions of this paper are threefold:

- We propose a principled approach to integrate KGE into the DLRS, which can fully exploit the value of information sharing between implicit semantics in the user-item interaction data and explicit semantics in the knowledge graph. The design of CIS, KEN and DRN layer is highly flexible and modular so that new KGE methods can be easily adapted.
- We design the CIS layer to realize item information sharing between the user-item implicit interaction matrix and the knowledge graph relation matrix, which can alleviate the data sparsity problem and obtain additional valuable information. Furthermore, we optimize the non-uniform sampler problem of traditional KG-based methods by two aggregations, namely hottest nodes sampler and k-largest node sampler.
- We perform extensive experiments on three public datasets to evaluate the effectiveness and characteristics of our proposed model. More importantly, we explore the boundary of KG-based methodology in real industrial-level online recommendation tasks, opposing to small-scale benchmark datasets.

2. Related work

2.1. Deep recommender systems

Due to the great success, DLRS gains much attentions in the research field of recommendation [1]. The DLRS with side information, which boasts greater potential to alleviate the data sparsity issue, are more relevant to this work. These approaches can be classified by the adopted type of information. Firstly, user and item's attribute information can be exploited in addition to the behavior signals. For instance, [3] proposed a hybrid model to alleviate the data sparsity issue, which utilized both the rating matrix and side information (i.e. attributes of users and items) and combined aSDAE and *matrix factorization* together. Secondly, KG information can be incorporated into the recommendation model. Methods which proposed recently include the RKGE model [13], the DKN model [22], the RippleNet model [23], the MKR model [21], the RuleRec model [19], and the KTUP model [14]. The third type of side information, namely, off-topic information, which is usually seen as unrelated to the recommendation tasks, can also be utilized. In [19] a framework is developed to capture features from user's off-topic content information (i.e. social tweets) and introduce them into *matrix factorization* based algorithms, which proves that even off-topic content information can be quite useful for recommendation performance improvement.

[1] https://developers.google.com/knowledge-graph/
[2] https://searchengineland.com/library/bing/bing-satori
[3] https://tupu.baidu.com/xiaoyuan/
[4] https://baike.sogou.com/v66616234.htm

2.2. Knowledge enhanced recommender systems

Introducing KG as a type of auxiliary information can effectively solve the data sparsity problem, and a number of knowledge enhanced recommender systems have been proposed recently [26]. In [27], the KG is embedded with classic methods like TransE [28] or enhanced methods like CACL [29], MultiE [30], IterE [31] and the entity embeddings are fed into an RNN-based sequential recommender. DKN [22] trains the entity and relation embeddings by learning knowledge graph features and then the entity embeddings are incorporated into the DLRS to learn the prediction function. Both the work above train the KGE in a separate way, which can be unfavorable in some situations as the KGEs are not optimized for specific tasks. The CKE [32], KGCN [24] and RippleNet [23] are designed to combine the recommendation model with the KGE learning in an end-to-end framework. However, they couldn't achieve effective information sharing between the explicit semantics from the KG and the implicit semantics from the behavior signals. The MKR model [21] regards recommendation modeling and KGE as two separate but related tasks for which thereby a multi-task learning framework is developed. The drawback is that the KGE module is not directly adapted to the recommendation task which may cause suboptimal performance. The major difference between our work and the above literatures is that we can offer a new perspective for recommender systems with the assistance of a heterogeneous knowledge graph.

3. The DKEN framework

In this section, we give the formulation of the research problem and the details of our approach.

3.1. Problem formulation

A recommendation model with KG enhancement can be formulated as follows. Given a user set $U = \{u_1, u_2, \ldots, u_{|U|}\}$ and an item set $V = \{v_1, v_2, \ldots, v_{|V|}\}$, we can define the user-item implicit feedback matrix as $Y = \{y_{uv} | u \in U, v \in V\}$, where y_{uv} is an indicator function of whether user u interactions with item v. The implicit feedbacks can be clicks, music/movie plays, payments, etc., depending on the application scenario and the business objectives. In addition to the user-item interaction matrix Y, a knowledge graph $\mathcal{G} = (E, R, E)$, where $E = \{e_1, e_2, \ldots, e_{|E|}\}$ is a set of entities, $R = \{r_1, r_2, \ldots, r_{|R|}\}$ is a set of relations, $|E|$ and $|R|$ represent the number of entities and relations in KG respectively. While $\mathcal{G} \subseteq E \times R \times E$ represents the set of triples, which is generally represented as (h, r, t), here $h \in E, t \in E$, and $r \in R$, where h and t represent head and tail entities, and r represents the relation between h and t. With the matrix Y and knowledge graph $\mathcal{G}$, the model is to predict the probability of a user u would interact with an unseen item v, which can be defined as a prediction function $\hat{y}_{uv} = \mathcal{F}(u, v; \Theta, Y, \mathcal{G})$, where Θ denotes the model parameters of function $\mathcal{F}$.

3.2. Framework

The model architecture of DKEN is illustrated in Fig. 1. It consists of four components, namely, the CIS Layers, the DNN Layer, the KEN Layer, and the DRN Layer. The DKEN model takes a user u, an item v, and a knowledge triple set $\mathcal{G}$ as input,

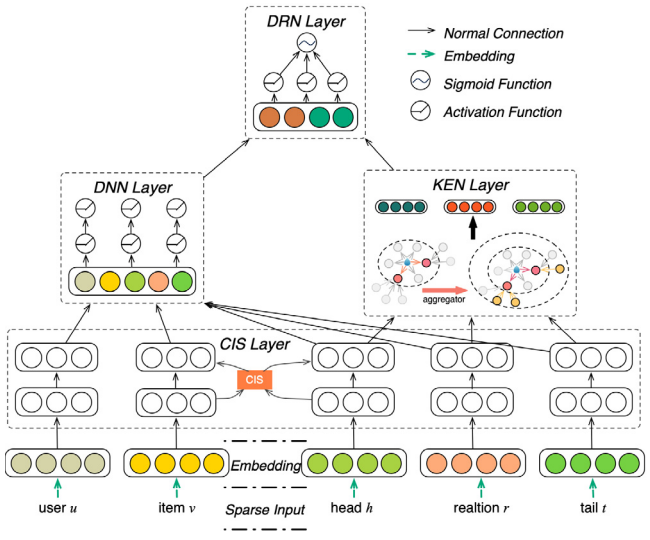


Fig. 1. Overview of the Proposed DKEN Framework. It is mainly composed of four parts: CIS Layers, DNN Layer, KEN Layer, and DRN Layer.

and outputs the predicted probability that user u will click item v. The CIS Layer is used to achieve information sharing and complementarity between explicit and implicit features. The DNN Layer on the left is a feed-forward neural network, which is used to learn high-order feature interactions. The KEN Layer on the right part, featuring two enhanced sampling algorithms, is used to learn the embeddings for the entities and relations in the KG. The DRN Layer combines high-order representation features from DNN Layer and the KGE features from KEN Layer into an end-to-end process to implement ensemble learning.

3.3. Cross-information sharing layer

To alleviate the data sparsity problem and obtain additional information, the CIS layer is designed to realize information sharing between item embeddings and entity embeddings, as depicted in Fig. 2, which are actually two types of representations for the same object.

Inspired by SoRec [33], we design a method of shared embedding to achieve information sharing. Our method can be described as:

$$\mathcal{L}_{CIS} = \sum_{Y_{i,j\neq 0}} l(Y_{i,j}, F_{i,j}) + \alpha_\eta \sum_{i,k} l(R_{i,k}, P_{i,k}) + \alpha_r \varphi(\theta) \tag{1}$$

Algorithm 1 Learning parameters for CIS Layer

Require: Interaction matrix Y, knowledge graph $\mathcal{G}$
Ensure: Prediction function $\mathcal{F}(u, v; \Theta, Y, \mathcal{G})$
1: Initialize all parameters
2: **for** number of training iteration
3: **for** t steps
4: Randomly draw a rating $Y(v_i, v_j, F_{i,j})$ and $\mathcal{G}(v_i, v_k, P_{i,k})$;
5: Update parameters of $\mathcal{F}$ by gradient descent;
6: **end for**
7: Update the learning rate α_η;
8: **end for**

In Eq. (1), the first term is set for the user-item implicit interactions matrix Y. While $Y_{i,j}$ indicates the scoring of item v_i by user u_j, in this paper it actually means that user u_j has clicked on item v_i. $F_{i,j}$ represents the correlation function between item v_i item v_j through user's historical click behavior, such as $F_{i,j} = f(V_i, V_j) = V_i^T V_j$, which is usually calculated based on Cosine similarity, Pearson similarity or other similarity measurement methods. The second term is designed for the knowledge graph relation R. $R_{i,k}$ represents the correlation between item i and entity k, and it should be noted that items are also entities in KG. $P_{i,k} = h(V_i, E_k)$ represents a function for correlation between entities through the item feature vector V_i and entity (tail) feature vector E_k. Parameter α_η is the learning rate used to control the iteration step size. The last term is the regular constraint of the variables contained in the above items, $\theta = \{\{V_i, E_j, \ldots\}, i = 1, 2, \ldots, n; j = 1, 2, \ldots, m\}$. In general, constrained regular terms can be described by different norms, such as $L_1, L_2, L_p, (0 \leqslant p < 1)$, etc. Parameter α_r is ordinarily used to balance the contribution of the above two items. $\sum_{Y_{i,j\neq 0}} l(Y_{i,j}, F_{i,j})$ and $\sum_{i,k} l(R_{i,k}, P_{i,k})$ can be expressed as follows:

$$\sum_{Y_{i,j\neq 0}} (Y_{i,j} - F_{i,j})^2 = \sum_{Y_{i,j\neq 0}} \left(Y_{i,j} - g\left(V_i^T V_j\right)\right)^2 \tag{2}$$

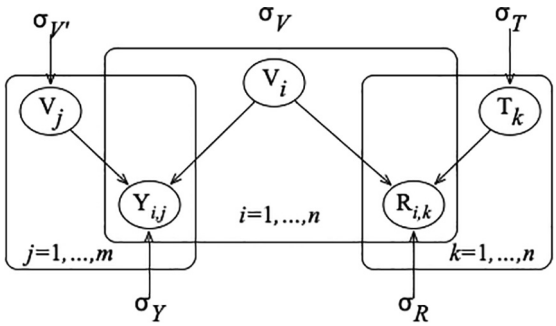


Fig. 2. Illustration of the information sharing process between item and head (tail), σ represents the hyper-parameters.

$$\sum_{S_{i,k}\neq 0}\left(R_{i,k}-P_{i,k}\right)^2=\sum_{S_{i,k}\neq 0}\left(R_{i,k}-g\left(V_i^T E_k\right)\right)^2 \tag{3}$$

The function $g(x)$ is the logistic function $g=1/(1+exp(x))$, which makes it possible to bound the range of $V_i^T E_k$ within the range [0,1]. Finally, CIS layer can be regarded as a pre-training phase in DKEN, and the loss function can be written as:

$$\mathcal{L}_{CIS}=\sum_{Y_{i,j}\neq 0}\left(Y_{i,j}-g\left(V_i^T V_j\right)\right)^2+\alpha_\eta\sum_{R_{i,k}\neq 0}\left(R_{i,k}-g\left(V_i^T E_k\right)\right)^2+\alpha_r||\theta||_F^2 \tag{4}$$

3.4. DNN layer

In view of the lack of high-level feature learning ability [22,34,35], we try to integrate all output embeddings of CIS layer into a DNN layer for high-level feature learning to enhance the model's generalization ability, as the left part is shown in Fig. 1. Denote the output of CIS layer as:

$$a^{(0)}=[e_U,e_V,e_H,e_R,e_T] \tag{5}$$

where e represents the embedding of U,V,E,R,T respectively. Then, $a^{(0)}$ is fed into the deep neural network, and the forward process is:

$$a^{(l+1)}=\sigma\left(W^{(l)}a^{(l)}+b^{(l)}\right) \tag{6}$$

where l is the layer depth and σ is an activation function. $a^{(l)}, W^{(l)}, b^{(l)}$ are the output, model weight, and bias of the l-th layer. The loss function can be written as $\mathcal{L}=\lambda_*||W||$, where λ_* denotes the regularization term and W denotes the set of parameters.

3.5. Knowledge-enhanced network layer

To define the user preference propagation on the KG, we use the framework proposed by RippleNet [23]. The preference propagation starts with an entity and then moves outward along the edge hop by hop and the signal intensity decays gradually in the process of propagation, which is just like a raindrop falling into a calm lake, with the waves producing layer after layer of ripples. As shown in Fig. 3, the color of the node gradually fades from dark to light, indicating the gradually decreasing weight. To characterize users' hierarchically extended preferences in terms of KG, [23] recursively define the set of k-hop relevant entities for user u as follows: Given interaction matrix Y and knowledge graph $\mathcal{G}$, the set of k-hop relevant entities for user u is defined as:

$$\mathcal{E}_u^k=\left\{t|(h,r,t)\in\mathcal{G},h\in\mathcal{E}_u^{k-1}\right\},k=1,2,\ldots,H \tag{7}$$

where $\mathcal{E}_u^0=\mathcal{V}_u=\{v|y_{uv}=1\}$ denotes a set of items that the user has ever clicked in the past, which can be seen as the seed set of user u in KG. The k-hop ripple set of user u is defined as the set of knowledge triples starting from $\mathcal{E}_u^{k-1}$:

$$\mathcal{S}_u^k=\left\{(h,r,t)|(h,r,t)\in\mathcal{G},h\in\mathcal{E}_u^{k-1}\right\},k=1,2,\ldots,H \tag{8}$$

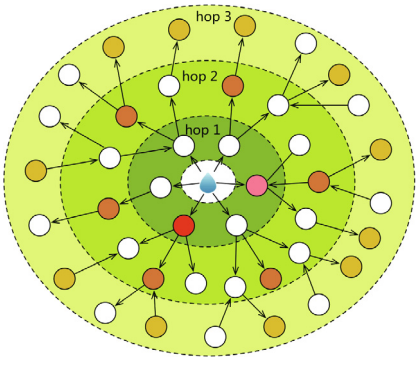


Fig. 3. Illustration of ripple sets of one raindrop in KG. The different ellipses represent the ripple sets with different hops. The fading green indicates weakening relation between the center raindrop and surrounding entities.

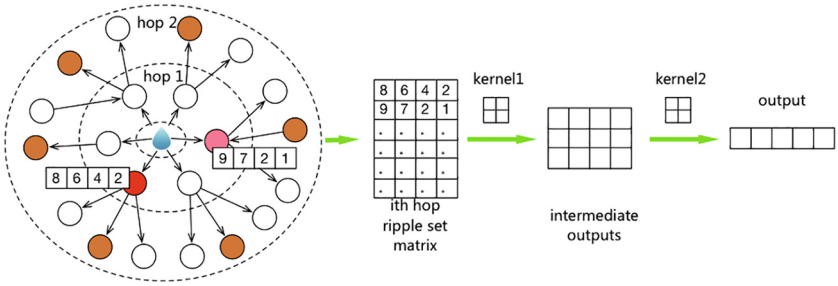


Fig. 4. Illustration of k-largest node sampler, which contains two CNN layers. We consider a central entity with 6 tails. Each tail gets the top 4 relation entities, represented by a 4-element vector, and each element represents the outdegree for the corresponding node. We can take a fixed-size set of neighbors and use a 2-D CNN to produce a new vector representation of fixed-size features for the central entity. It can be seen from the figure that central entity includes 6 neighbors, and it takes the top 4 relation entities for each tail. The 4 largest values can be selected from the neighbors in descending order according to their outdegree. For example, a 4-element vector of the first tail is $(8, 6, 4, 2)$, whose color is red. By repeating the same process for other tails, we obtain fixed-size 4-element vectors. Concatenating them gives a 2-D data of grid-like structure, which is 2×2 kernel. Afterwards, a 2-D CNN is applied to generate the final vector. We can use any CNN model with different kernel sizes to ensure that the final output is a vector.

To optimize the non-uniform sampler problem of some state-of-the-art methods, we have designed the KEN layer, as the right part is shown in Fig. 1. *Hottest nodes sampler* and *k-largest node sampler* are designed to obtain the user's principal preference signal during the performance propagation operations.

Hottest node sampler uses the max-pooling method to extract the hottest nodes and find each hottest child node successively. As is displayed in Fig. 3, the color of the node gradually fades from dark to light, it indicates the gradual decrease of the weight. Given an outdegree set $O = \{o_1, o_2, \ldots, o_{|O|}\}$, where o represents the outdegree of neighbor node, the k hop set of each node is:

$$\begin{aligned} &\mathcal{S}_u^{top\,m} = \left\{(h, r, t_o)|(h, r, t) \in \mathcal{G}, h \in \mathcal{E}_u^{k-1}\right\}, \\ &k = 1, 2, \ldots, H; o \geqslant k\_max(O, m) \end{aligned} \tag{9}$$

where $k\_max(O, m)$ is a bivariate function to calculate the m-th largest value from the outdegree set O, and m is the triple memory size. During ripple set sampling, we sort each neighbor node in descending order according to its outdegree and then take a fixed-size set of neighbors instead of a random ripple set to further extract salient information.

Besides, k-largest node sampler with the way of *convolution neural network* (CNN) can also aggregate new feature representation from the neighbor outdegree vector of center nodes, and transform these vector features into grid-like structured matrix with combined the outdegree vector of center node. After two layers of CNN, the transformed vector is used as the final vector of this sample. The detailed implementation process is shown in Fig. 4. In a ripple set, the blue raindrop is the center node, and the other nodes are the neighbor nodes.

3.6. DRN layer

The DRN Layer also leverages multiple layers to learn high-order feature interactions of DNN layer and KGE features of KEN layer into the end-to-end process automatically as depicted in Fig. 1. All parameters and the network parameters $\left(W^{(l)}, b^{(l)}\right)$ are trained jointly for the combined prediction model:

$$\hat{y}_{uv} = sigmoid(y_{DNN} + y_{KEN}) \tag{10}$$

where $\hat{y}_{uv} \in (0, 1)$ is the predicted CTR, y_{DNN} the output of DNN layer, and y_{KEN} the output of KEN layer.

The complete loss function of DRN is as follows:

$$\mathcal{L} = \mathcal{L}_{DRN} + \mathcal{L}_{KEN} + \mathcal{L}_{REG} = \sum_{u \in U, v \in V} \mathcal{T}\left(\hat{y}_{uv}, y_{uv}\right) + \frac{\omega}{2} \sum_{r \in R} ||I_r - E^T R E||_2^2 + \frac{\lambda}{2} \left(||\theta||_2^2 + \sum_{r \in R} ||R||_2^2 \right) \tag{11}$$

In Eq. (11), the first term is the cross-entropy loss between ground truth of interactions Y and predicted value by DRN, where U and V represent the set of users and items respectively. The second term is the KGE term, which is used to calculate the squared error, where I_r is the slice of the indicator tensor I in KG and $E^T R E$ is the reconstructed indicator matrix. The last item is aimed to prevent model over-fitting by introducing λ, where $\theta = \{W, V, E\}$, and R is the embedding matrix of relation r.

We use *stochastic gradient descent* (SGD) algorithm to optimize the loss function. The learning algorithm of CIS layer and DKEN are presented in Algorithm 1 and Algorithm 2. The following [36], we adopt a negative sampling strategy in each training iteration to improve the computational efficiency. Then, we can calculate the gradient of loss $\mathcal{L}$ with respect to model parameter Θ, and update all parameters by back propagation based on a minibatch of samples.

Algorithm 2 Learning parameters for DKEN

Require: Interaction matrix Y, knowledge graph $\mathcal{G}$
Ensure: Prediction function $\mathcal{F}(u, v; \Theta, Y, \mathcal{G})$
1: Initialize all parameters
2: Pretrain CIS layer with Algorithm 1
3: Calculate ripple sets $\left\{\mathcal{S}_u^k\right\}_{k=1}^{H}$ for each user u
4: **for** number of training iteration
5: **for** t steps
6: Sample minibatch of positive and negative interactions from Y;
7: Sample $e \sim \mathcal{S}(v)$ for each item v in the minibatch;
8: Update parameters of $\mathcal{F}$ by gradient descent;
9: **end for**
10: Sample minibatch of true and false triples from $\mathcal{G}$;
11: Calculate gradients $\frac{\partial \mathcal{L}}{\partial U}, \frac{\partial \mathcal{L}}{\partial V}, \frac{\partial \mathcal{L}}{\partial E}, \left\{\frac{\partial \mathcal{L}}{\partial R}\right\}_{r \in R}$ on the minibatch;
12: Update $U, V, E, \{R\}_{r \in R}$ by gradient descent with the learning rate η ;
13: **end for**

4. Links to existing work

In this paragraph, the differences with related work are summarized through the comparison between state-of-the-art methods with the proposed one. First of all, it is different from MKR [21], RuleRec [19] due to the adopted feature matrix sharing representation in CIS layer, which focuses more on MF-based social recommendation methods in prior works, in order to realize item information sharing between the user-item implicit interaction matrix and the knowledge graph relation matrix, i.e. the item feature matrix is used as the intermediary of the sharing representation to associate the score information with the relation information. What's more, compared with attention-based aggregation, which employs the neural attention mechanism KGAT [37], KGCN [24] to learn the weight of each neighbor during a propagation, we introduce *hottest nodes sampler* and *k-largest node sampler* to optimize the non-uniform sampler problem to explore users' potential interests in KEN layer. In particular, to share information between implicit semantics information in DLRS and explicit semantics information from the knowledge graph, DKEN attempts to provide a new perspective to respond to the above issues and has been applied in the industrial recommendation business successfully.

5. Experiments

In this section, we perform experiments on three public datasets and an industrial online application to evaluate our proposed method. We aim to answer the following research questions:

- **RQ1:** Compared with the state-of-the-art KG-enhanced methods, how does our method perform?
- **RQ2:** How the different choices of hyper-parameters (e.g., different components and the sampler method in KEN layer) affect the performance of DKEN?
- **RQ3:** Can DKEN provide reasonable explanations about user preferences towards items and get better recommendations in the industrial datasets?

5.1. Experimental setup

5.1.1. Public datasets

The following three datasets were used in our experiments for movie, book and music recommendation.

- **MovieLens-1 M**[5] is a well-known dataset for movie recommendation, which contains approximately 1 million ratings (ranging from 1 to 5) from 6040 users on 3706 movies.
- **Book-Crossing**[6] is a widely used benchmark dataset in book recommendation, which contains about 1.1 million ratings (ranging from 1 to 10) of 270,000 books by 90,000 users.

[5] https://grouplens.org/datasets/movielens/1m/
[6] http://www2.informatik.uni-freiburg.de/cziegler/BX/

Table 1
Detailed statistics of the three public datasets.*#edge types* indicates the number of relation types, *#1-hop triples* indicates the number of 1-hop relevant entities for users, and *#2-hop triples* denotes the number of 2-hop relevant entities for users, and *#sparsity level* shows the sparsity of each dataset (i.e., *#sparsity level* = 1-*#interactions*/ (*#users* × *#items*)).

	MovieLens-1 M	Book-Crossing	Last.FM
#users	6036	17860	1872
#items	2445	14967	3846
#interactions	753772	139746	42364
#edge types	12	25	60
#entity	182011	77903	9366
#relationship	2483990	303000	31036
#1-hop triples	20782	19876	15518
#2-hop triples	178049	65360	77930
#sparsity level	0.948925	0.999447	0.994116

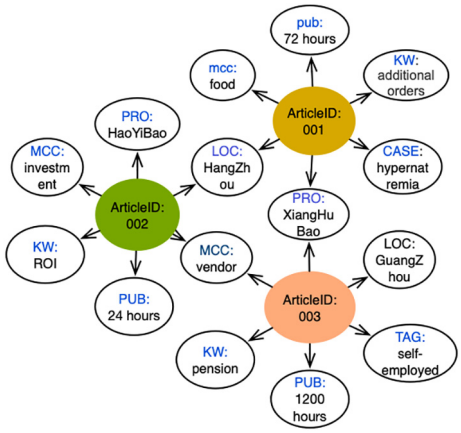


Fig. 5. Illustration of feeds recommendation KG, which contains 3 articles. Different articles are linked through entities, and each article is centered on an article ID.

- **Last.FM**[7] dataset is collected from an online music website, which contains several musician listening information from two thousand users.

Some data preprocessing work should be done before modeling. For the MovieLens-1M dataset, we treat ratings 4 and 5 as positive feedbacks, and use all ratings and listening count as positive signals in Book-Crossing and Last.FM. We use the Microsoft Satori knowledge engine to extract knowledge triples and construct the KG for the three datasets. The detailed statistics are summarized in Table 1.

5.1.2. Alipay feeds recommendation dataset

We have also evaluated the models on a Alipay feeds dataset, which comes from an industrial online recommendation application in the Ant Financial Service Group. The dataset contains interactions of over 2.9 million users on over 5 thousand items. The sample is labeled positive for click actions and negative otherwise. We got more than 9 thousand entities and 12 types of relations (e.g., KW, CATE, PUB, LOC, and MCC) in the same way as above for KGE construction. The KG demo of insurance domain is shown in Fig. 5.

5.1.3. Baselines

We have evaluated our approach with the following baselines for performance comparison:

- **LibFM** [38] combines the generality of feature engineering with a factorization model for recommendation.
- **PER** [39] combines heterogeneous relationship information from different users and items with a heterogeneous information network to do the recommendation.
- **Wide&Deep** [34] is a well-known recommendation model, which combines the logistic model with the DNN network.

[7] https://grouplens.org/datasets/hetrec-2011/

- **DKN** [22] uses a multi-channel and word-aligned knowledge-aware CNN to fuse the semantic level and knowledge level representation for news recommendation.
- **RippleNet** [23] expands users' potential interest according to the KG and stimulates user preference propagation on the knowledge graph.
- **MKR** [21] is a multi-task feature learning approach for recommendation, which utilizes a KGE task to assist recommendation task.
- **KGCN** [24] captures inter-item relatedness by mining their associated attributes on the KG and learns uses' potential long-distance interests.
- **KGN-DNN** adds a DNN network on the CIS layer model for higher-order feature learning as Fig. 1 shown.

5.1.4. Evaluation scheme

We use AUC and Accuracy (ACC) to evaluate the performance of different models for offline implicit feedback prediction, and PVCTR (click through rate on page views) and UVCTR (click through rate on unique user views) for online evaluation. We randomly divide each dataset into the training set, validation set, and test set, accounting for 60%, 20%, 20%, respectively. Each experiment is repeated 5 times, and the average performance is reported. The optimal hyper-parameters settings for these datasets are shown in Table 2, where d denotes the latent dimension, H denotes the number of maximum hop, ω denotes the weight of KGE, λ denotes the $L2$ regularization, m denotes the size of the ripple set, and η denotes the learning rate. Note that for fair consideration, the latent dimensions of other compared baselines are set the same in Table 2, while other hyper-parameters of baselines are chosen by grid search.

5.2. Performance Comparison on public datasets (RQ1)

Table 3 shows the results of different models in CTR (Click Through Rate) prediction experiments. We discuss our findings as follows.

- LibFM and Wide&Deep perform better than PER and DKN, which shows that they can make full use of the knowledge from knowledge graph and achieve better performance.
- PER and DKN perform unsatisfactorily on movie, book and music recommendation. PER cannot combine heterogeneous relationship information and use heterogeneous information network effectively. However, the entity embeddings, which are required in advance of using DKN, cannot participate in the end-to-end way of the training process.
- RippleNet and KGCN are more sensitive than MKR and DKEN for the sparsity of datasets owing to the worse performance on book and music recommendation. The sparsity level of movieLens-1M, Book-Crossing, and Last.FM are 0.948925, 0.999447, and 0.994116, respectively. DKEN achieves average AUC gains of 6.00% and 4.13%, as well as ACC gains of 8.85% and 7.84% in book and music recommendation. But it achieves AUC gains of 1.53% and ACC gains of 1.55% in movie recommendation, which can prove that DKEN has better performance on more sparse datasets.

Table 2
Detailed hyper-parameters settings for all datasets.

Dataset	Hyper-parameters Settings
MovieLens-1 M	$d=32, H=2, \omega=10^{-3}, \lambda=10^{-7}, \eta=0.005, m=8$
Book-Crossing	$d=16, H=3, \omega=10^{-3}, \lambda=10^{-7}, \eta=0.0007, m=8$
Last.FM	$d=16, H=3, \omega=10^{-3}, \lambda=10^{-7}, \eta=0.0007, m=32$
Feeds Recommendation Data	$d=8, H=4, \omega=10^{-4}, \lambda=10^{-3}, \eta=0.001, m=32$

Table 3
The results of AUC and ACC in CTR prediction on the three public datasets.

Model	MovieLens-1 M		Book-Crossing		Last.FM	
	AUC	*ACC*	*AUC*	*ACC*	*AUC*	*ACC*
LibFM	0.892(−4.1%)	0.812(−4.8%)	0.685(−7.8%)	0.640(−9.1%)	0.777(−4.5%)	0.709(−6.8%)
PER	0.710(−23.7%)	0.664(−22.2%)	0.623(−16.2%)	0.588(−16.5%)	0.633(−22.2%)	0.596(−21.7%)
Wide&Deep	0.898(−3.4%)	0.820(−3.9%)	0.712(−4.2%)	0.624(−11.4%)	0.756(−7.1%)	0.688(−9.6%)
DKN	0.655(−29.6%)	0.589(−30.9%)	0.622(−16.3%)	0.598(−15.1%)	0.602(−26.0%)	0.581(−23.7%)
RippleNet	0.913(−1.8%)	0.835(−2.1%)	0.729(−1.9%)	0.662(−6.0%)	0.768(−5.7%)	0.691(−9.2%)
MKR	0.917(−1.4%)	0.843(−1.2%)	0.734(−1.2%)	**0.704**	0.797(−2.1%)	0.752(−1.2%)
KGCN	0.919(−1.2%)	0.845(−0.9%)	0.675(−9.2%)	0.620(−11.9%)	0.796(−2.2%)	0.721(−5.3%)
KGE−DNN	0.923(−0.8%)	0.850(−0.4%)	0.740(−0.4%)	0.696(−1.1%)	0.812(−0.2%)	0.759(−0.3%)
DKEN	**0.930**	**0.855**	**0.743**	0.697(−1.0%)	**0.814**	**0.761**

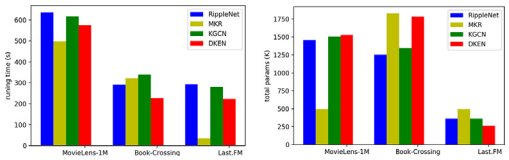


Fig. 6. (a) Running time of all methods on the three public dataset. (b) Total params of all methods on the three public dataset.

- KGE-DNN [DKEN without KEN layer] based on CIS layer performs better than RippleNet and KGCN on movie, book and music recommendation, which demonstrates that it can learn higher-order feature interactions, and capture user interest through information sharing between items and entities in the embedding phase.
- In general, our DKEN model performs best among all models on the three public datasets. It achieves average AUC gains of 12.18%, 9.25% and 12.38%, and ACC gains of 12.38%, 10.34% and 13.36% in movie, book and music recommendation, respectively. This demonstrates the effectiveness of DKEN on better exploiting both implicit semantics in the behavior data and the explicit semantics in the knowledge graph.
- To analyze the performance of DKEN, we compare it with several state-of-the-art baselines in their optimal hyper-parameters from two common ways (running time and total parameters)[40]. As can be seen from Fig. 6, the proposed model performs well both in terms of running time and total parameters.

5.3. Parameter sensitivity (RQ2)

In this section, we analyze the influence of different modular layers in our model, and examine the hyper-parameters d,m, ω, and H of DKEN on three public datasets.

5.3.1. Impact of CIS layer, DNN layer and KEN layer

Experimental results in Table 4 show the detailed performance of different layers in different datasets. In MovieLens-M, the ERN layer performs best, while the DNN layer plays little role in the DKEN model. However, the CIS layer performs best on Book-Crossing and Last.FM, which also proves that CIS can alleviate the data sparsity problem and obtain additional valuable information.

5.3.2. Impact of the sampler method in KEN layer

Compared with other aggregation, we have designed two non-uniform samplers during preference propagation to better explore users' potential interests and improve the performance on KG. As shown in Fig. 7, DKEN with k-largest node sampler performs better than the model with a random sampler, which demonstrates that the developed k-largest node sampler method learns more useful information and captures user interest than random sampler.

5.3.3. Impact of embedding dimension

We have modulated the size of embedding dimension to further validate the efficiency of embedding. From Table 5 we can draw the observation that the performance is improving with the increase of embedding dimension while downgrading after the critical point is reached, and the DKEN model achieves the best performance when d is 16 or 32. As the dimension increases, the representation power of the model improves. However, at the same time, more noise is involved and the model begins to suffer from overfitting.

5.3.4. Impact of the size of ripple set for each hop

We have investigated the efficiency about the size of the ripple set for each hop. Table 6 shows that the AUC metrics improve as the ripple size increases at the beginning but downgrade after the critical point is reached. This can be explained in the similar way as the impact of the embedding dimension.

Table 4
The detailed performance of different layers.

	MovieLens-1 M		Book-Crossing		Last.FM	
	AUC	*ACC*	*AUC*	*ACC*	*AUC*	*ACC*
DKEN(without CIS layer)	0.926	0.851	0.725	0.671	0.793	0.733
DKEN(without KEN layer)	0.923	0.850	0.741	0.696	0.794	0.747
DKEN(without DNN layer)	0.929	**0.857**	0.740	0.696	0.812	0.759
DKEN	**0.930**	0.855	**0.743**	**0.697**	**0.814**	**0.761**

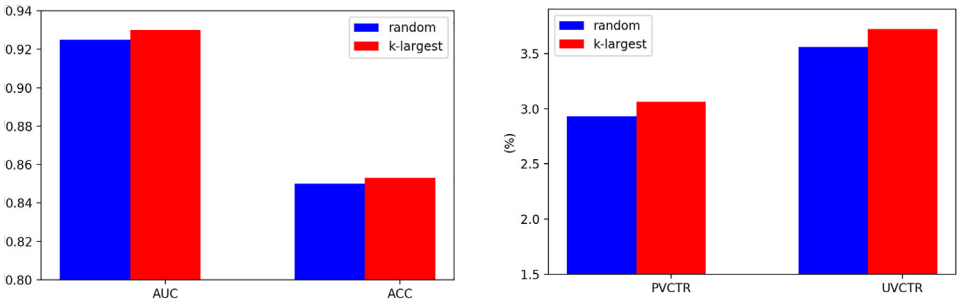


Fig. 7. (a) The AUC and ACC with different sampling methods on MovieLens-1M dataset. (b) The PVCTR and UVCTR of different sampling methods with an industrial online AB test.

Table 5
AUC result of DKEN with diffrerent dimensions of embedding d.

d	MovieLens-1 M		Book-Crossing		Last.FM	
	AUC	*ACC*	*AUC*	*ACC*	*AUC*	*ACC*
2	0.909	0.831	0.734	0.693	0.796	0.758
4	0.917	0.840	0.736	0.694	0.809	0.759
8	0.927	0.854	0.739	0.693	0.813	0.760
16	0.928	**0.855**	**0.743**	**0.697**	**0.814**	**0.761**
32	**0.930**	0.853	0.740	0.694	0.810	0.757
64	0.928	0.853	0.741	0.696	0.812	0.759

Table 6
AUC result of DKEN with diffrerent size of ripple set for each hop m.

m	MovieLens-1 M		Book-Crossing		Last.FM	
	AUC	*ACC*	*AUC*	*ACC*	*AUC*	*ACC*
2	0.926	0.852	0.741	0.694	0.812	0.755
4	0.927	0.853	0.740	0.696	0.813	0.757
8	**0.930**	0.853	**0.743**	**0.697**	0.812	0.758
16	0.929	0.855	0.741	0.695	0.812	0.759
32	**0.930**	**0.857**	0.741	**0.697**	**0.814**	**0.761**
64	0.929	0.855	0.740	0.695	0.810	0.758

5.3.5. Impact of the weight of the KGE term

We have investigated the efficiency about the weight of KGE (i.e. the parameter ω in Table 2) term by modulating the weight from 0.00001 to 0.01. Table 7 shows the results of AUC in three public datasets, and DKEN model performs best when the weight of KGE is 0.0001. The results also show that DKEN can not predict the CTR with a small weight of KGE term due to the slight regularization constraints, but a large weight may mislead an objective function.

5.3.6. Impact of the number of the maximum hop

We have investigated the performance about the number of the maximum hop. As can be concluded from Table 8, the performance is improving with the increase of hop number, and downgrading after the critical point is reached. A small H is difficult to explore the long-distance correlations and dependencies of entities, while a large H may bring more noises from useful information. Meanwhile, in MovieLens-1M, the best performance is achieved when H is 2 and others are achieved when H is 3, which shows that a sparse data may require a large H, but a dense data just requires a smaller H.

5.4. Alipay feeds recommendation dataset and online experiment (RQ3)

5.4.1. Performance comparison

We have also evaluated different algorithms on a feeds recommendation dataset from an industrial online application. The comparison results are reported in Table 9 and the detailed results of online metrics are shown in Fig. 8, from which we can conclude that:

Table 7
AUC result of DKEN with diffrerent weight of the KGE term ω.

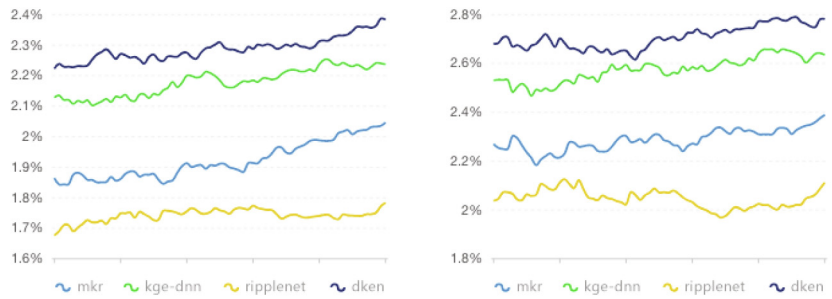
ω	MovieLens-1 M		Book-Crossing		Last.FM	
	AUC	*ACC*	*AUC*	*ACC*	*AUC*	*ACC*
0.00001	0.828	0.854	0.742	0.697	0.813	0.757
0.00005	0.929	0.855	0.742	0.696	0.813	0.758
0.0001	**0.930**	0.853	**0.743**	**0.697**	**0.814**	**0.761**
0.0005	0.929	0.855	0.741	0.695	0.811	0.758
0.001	0.928	0.854	0.740	0.693	0.812	0.757
0.005	0.929	**0.856**	0.740	**0.697**	0.812	0.758
0.01	0.924	0.848	0.740	0.693	0.811	0.759

Table 8
AUC result of DKEN with diffrerent number of the maximum hop H.

H	MovieLens-1 M		Book-Crossing		Last.FM	
	AUC	*ACC*	*AUC*	*ACC*	*AUC*	*ACC*
1	0.927	0.852	0.741	0.694	0.813	0.757
2	**0.930**	0.853	0.740	0.696	0.812	0.758
3	0.929	**0.855**	**0.743**	**0.697**	**0.814**	**0.761**
4	0.928	0.854	0.740	0.693	0.811	0.759

Table 9
The results of different models on the feeds recommendation dataset.

	PVCTR	UVCTR	AUC	ACC
RippleNet	1.69%	2.04%	0.615	0.692
MKR	1.84%	2.27%	0.618	0.679
KGE-DNN	2.14%	2.51%	0.625	0.748
DKEN	**2.24%**	**2.67%**	**0.637**	**0.760**



(a) PVCTR metric (b) UVCTR metric

Fig. 8. (a) The detail performance of the PVCTR metric at different time periods of one day. (b) The detail performance of the UVCTR metric in a whole day.

- Experimental results in Table 9 have indicated that the DKEN model outperforms the RippleNet, MKR, and KGE-DNN by yielding 32.54%, 21.74% and 4.67% of the PVCTR metric, 30.88%, 17.62% and 6.37% of the UVCTR metric, respectively. DKEN also performs the best in terms of AUC and ACC. These results demonstrate that our DKEN model is highly effectiveness for online recommendations.
- KGE-DNN [DKEN without KEN layer] based on CIS layer performs better than MKR and RippleNet, which shows that KGE-DNN can learn higher-order features better than MKR.
- Fig. 8 shows the PVCTR and UVCTR of DKEN with other baselines at different time periods of one day, which draws the observation that the curve of DKEN is always higher than the baselines in a whole day and proves the efficiency of DKEN model. In addition, compared with other baselines, DKEN can be improved steadily, which indicates that DKEN is more stable and robust in online scenario.

Table 10
Illustration of click history for a randomly sampled user, the first column is the title of the content. The second column is the category distribution of articles, which is indicated as category:number of related articles, such as C013:4. The third column is the distribution of entities, indicated as entities: number of related articles, e.g. patients:5.

Click History	Categories	Entities of KG
1. Do you know the etiology of gouty arthritis and the methods of disease prevention? 2. What is the best way to preserve one's health during the Spring Festival? 3. The implementation plan of medical care, health, poverty alleviation and medical expenses coverage is closely related to everyone! 4. After 50 years old, you must give up these habits! Health is better than anything! 5. How important is it for children to have kid's insurance when the Spring Festival is at home?	{C013:4,C028:2, C025:1,C007:1, C005:1,C001:1}	{patients:5,disease:4,insurance:3, hospital:3, medicare:3,doctor:3, age:2,cardiovascular:2,claims:2, cold:2,woman:2, hypertension:2, TCM:2,chronic:2,man:2,medicaid:2, outpatient:2,parents:2, diabetes:2,sanitarian:2,treatment:2, middleaged:2, reimbursement:2}

Table 11
Illustration of the recommended results of KGE-DNN [DKEN without KEN layer], RippleNet, MKR and DKEN respectively, in which KGE-DNN does not employ KGE. Bold and italicized parts indicate the same categories and entities as the user's historical click content.

	Top 2 Recommended Records	Categories	Entities of KG
KGE-DNN	1. How can Alipay's outpatient reimbursement claim the DuoShouDuoBao in the merchant service? 2. If you are a merchant, how to make good use of Alipay and let us pay for your medical treatment?	C019 C007	{premium,***woman***,kids,pension,social-security,staff,endowment-insurance,aging} {medical-fee,hyperplasia,tympanitis, rheumatism,***woman***, medical, ***treatment***}
RippleNet	1. Can I continue to receive full reimbursement for outpatient service in 1000 RMB? 2. Endowment Insurance has changed! If you don't understand these problems, your social security will be paid for nothing!	C002 ***C001***	{sequela,reimbursement,palsy, transplantation,hepatitis, nephrosis, stroke,tumor,outpatient} {***insurance***,***woman***,kids,pension, endowment-***insurance***, ***medicaid***, staff,endowment-insurance, retirement,aging}
MKR	1. Open the gift bag! 80 million gift bags are waiting for you to collect! 2. The DouShouDuoBao of Alipay, Merchant will have free insurance if they collect money with the collection code.	C019 ***C001***	{family,medical-insurance,***age***, ***medicare***,***disease***,***medicaid***, oldness} {emergency,***medicare***,***reimbursement***, coverage,courier, ***insurance***, wife,***hospital***,merchant,university, couple,civilians, ***claims***}
DKEN	1. How to claim reimbursement for outpatient service? DuoShouDuoBao in Merchant Service. 2. The red envelope is smaller. DouShouDuoBao will make us full of happiness!	***C013*** ***C001***	{merchant,***claims,disease,treatment, medicare, hospital***,coverage,outpatient, emergency,civilians, ***reimbursement***, endowment-insurance,***insurance***} {***disease,insurance***,merchant,***treatment, medicare,claims,*** *medicaid*,coverage}

5.4.2. Case study

To demonstrate the effectiveness of using KG, we randomly chose 5 historical click records of a user, as illustrated in Table 10. We mainly analyze the results from two perspectives: category coincidence and entity coincidence [41]. If the coincidence of categories and entities between historical records and recommended results is higher, it is more consistent with the user's preference. Table 11 shows the recommended results of KGE-DNN [DKEN without KEN layer], RippleNet, MKR and DKEN respectively. By comparing categories and entities of recommended candidates, the results of KGE-DNN do not match users' historical content very well. Oppositely, we can clearly observe that the categories and entities of DKEN recommended candidates are the same as those historical records. Furthermore, the coincidence is more higher. To some extent, this approach can enhance the interpret-ability of recommendation objectives.

6. Conclusion and future work

In this paper, we study the problem of data sparsity for recommendation models. We propose DKEN, an end-to-end framework that takes both advantages of DLRS and KGE. With the DLRS module, it can learn higher-order feature interactions based on CIS layer. With the KGE module, it can learn representations for explicit semantics and the learnt representations can be optimized for a specific recommendation task in an end-to-end process. Specifically, the CIS layer achieves information sharing between implicit semantics and explicit semantics to be beneficial for the performance. The KEN layer with two effective samplers optimizes the non-uniform sampler problem during the performance propagation phase. Finally, we demonstrate the significant superiority of DKEN over baseline models through extensive evaluations on both public datasets and a real-world online scenario. More importantly, we propose an important research question and direction, namely how to design a model framework to combine the advantages of both the knowledge graph and the DLRS, and we think that it's valuable to attract more attention on this direction from both the research and industry communities.

For future work, we plan to (1) introduce heterogeneous network embedding to our framework to extract semantic representations from the KG more effectively, (2) explore constrained recommendation based on domain knowledge graph, which is a promising way to integrate domain knowledge with DLRS.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper. 1. The data used in this research does not involve any Personal Identifiable Information(PII). 2. The data used in this research were all processed by data abstraction and data encryption, and the researchers were unable to restore the original data. 3. Sufficient data protection was carried out during the process of experiments to prevent the data leakage and the data was destroyed after the experiments were finished. 4. The data is only used for academic research and sampled from the original data, therefore it does not represent any real business situation in Ant Financial Services Group.

CRediT authorship contribution statement

Xiaobo Guo: Conceptualization, Methodology, Software, Validation, Writing - original draft. **Wenfang Lin:** Conceptualization, Methodology, Software, Validation, Writing - original draft. **Youru Li:** Conceptualization, Methodology, Software, Validation, Writing - original draft. **Zhongyi Liu**: Review & editing. **Lin Yang:** Data curation. **Shuliang Zhao:** Visualization, Investigation, Supervision. **Zhenfeng Zhu:** Rreview & editing.

References

[1] S. Zhang, L. Yao, A. Sun, Deep learning based recommender system: A survey and new perspectives, arXiv:1707.07435..
[2] P. Covington, J. Adams, E. Sargin, Deep neural networks for youtube recommendations, Proceedings of the 10th ACM Conference on Recommender Systems, ACM, 2016, pp. 191–198.
[3] X. Dong, L. Yu, Y. Sun ZhonghuoWu, A hybrid collaborative filtering model with deep structure for recommender systems, in: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17), 2017, pp. 1309–1315.
[4] H. Wang, N. Wang, D.-Y. Yeung, Collaborative deep learning for recommender systems, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2015, pp. 1235–1244.
[5] L. Gao, H. Yang, Recommendation with multi-source heterogeneous information, in: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, 2018, pp. 3378–3384.
[6] J. Wang, P. Huang, H. Zhao, Z. Zhang, B. Zhao, D.L. Lee, Billion-scale com-modity embedding for e-commerce recommendation in alibaba, in: Proceedings of the 24th ACM SIGKDD In-ternational Conference on Knowledge Discovery & Data Mining, 2018, pp. 839–848.
[7] E. Palumbo, G. Rizzo, R. Troncy, entity2rec: Learning user-item relatedness from knowledge graphs for top-n item recommendation, in: Proceedings of the Eleventh ACM Conference on Recommender Systems, ACM, 2017, pp. 32–36.
[8] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 855–864.
[9] G. Semeraro, P. Lops, P. Basile, Knowledge infusion into content-based recommender systems, in: Proceedings of the Third ACM Conference on Recommender Systems, 2009, pp. 301–304.
[10] R. Xie, Z. Liu, J. Jia, H. Luan, M. Sun, Representation learning of knowledge graphs with descriptions, in: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, 2016, pp. 2659–2661.
[11] Y. Zhang, Q. Ai, X. Chen, P. Wang, Learning over knowledge-base embeddings for recommendation, in: arXiv:1803.06540(2018), 2018..
[12] G. Ji, K. Liu, S. He, J. Zhao, Knowledge graph completion with adaptive sparse transfer matrix, in: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, 2016, pp. 985–991.
[13] Z. Sun, J. Yang, J. Zhang, A. Bozzon, L.-K. Huang, C. Xu, Recurrent knowledge graph embedding for effective recommendation, in: Proceedings of the 12th ACM Conference on Recommender Systems, ACM, 2018, pp. 297–305.
[14] Y. Cao, X. Wang, X. He, Z. hu, T.-S. Chua, Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences, arXiv:1902.06236..
[15] R. Catherine, W. Cohen, Personalized recommendations using knowledge graphs: A probabilistic logic programming approach, in: Proceedings of the 10th ACM Conference on Recommender Systems, ACM, 2016, pp. 325–332.
[16] S. Guan, X. Jin, Y. Wang, X. Cheng, Shared embedding based neural networks for knowledge graph completion, in: Proceedings of the 27th ACM International Conference on Information and Knowledge Management(CIKM), 2018, pp. 247–256.
[17] J. Li, F. Xia, W. Wang, Acrec: A co-authorship based random walk model for academic collaboration recommendation, in: Proceedings of the 23rd International Conference on World Wide Web, 2014, pp. 1209–1214.
[18] F. Vasile, E. Smirnova, A. Conneau, Meta-prod2vec: Product embeddings using side-information for recommendation, in: Proceedings of the 10th ACM Conference on Recommender Systems, ACM, 2016, pp. 225–232.
[19] W. Ma, M. Zhang, Y. Cao, W. Jin, C. Wang, Y. Liu, S. Ma, X. Ren, Jointly learning explainable rules for recom-mendation with knowledge graph, in: Proceedings of the 2019 World Wide Web Conference, 2019, pp. 13–17.
[20] Y. Cao, X. Wang, X. He, Z. Hu, T.-S. Chua, Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences, International World Wide Web Conference(WWW'19) (2019).
[21] H. Wang, F. Zhang, M. Zhao, W. Li, X. Xie, M. Guo, Multi-task feature learning for knowledge graph enhanced recommendation, in: Proceedings of The 2019 Web Conference, 2019.
[22] H. Wang, F. Zhang, X. Xie, M. Guo, Dkn: Deep knowledge-aware network for news recommendation, in: Proceedings of the 2018 World Wide Web Conference on World Wide Web, 2018, pp. 1835–1844.
[23] H. Wang, F. Zhang, J. Wang, M. Zhao, W. Li, X. Xie, M. Guo, 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), 2018, pp. 417–426.
[24] H. Wang, M. Zhao, X. Xie, W. Li, M. Guo, Knowledge graph convolutional networks for recommender systems, International World Wide Web Conference (WWW'19) (2019).
[25] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2016) 504–507.
[26] Q. Wang, Z. Mao, B. Wang, L. Guo, Knowledge graph embedding: A survey of approaches and applications, IEEE Trans. Knowl. Data Eng. 29 (12) (2017) 2724–2743.

[27] H. Jin, Z. WayneXin, H. Dou, Improving sequential recommendation with knowledge-enhanced memory networks, in: Proceeding The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, 2018, pp. 505–514.
[28] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, in: Proceedings of the 26th International Conference on Neural Information Processing Systems, 2013, pp. 2787–2795.
[29] B. Oh, S. Seo, K.-H. Lee, Knowledge graph completion by context-aware convolutional learning with multi-hop neighborhoods, in: Proceedings of the 27th ACM International Conference on Information and Knowledge Management(CIKM), 2018, pp. 257–266.
[30] Z. Zhang, F. Zhuang, Z.-Y. Niu, D. Wang, Q. He, Multie: Multi-task embedding for knowledge base completion, in: Proceedings of the 27th ACM International Conference on Information and Knowledge Management(CIKM), 2018, pp. 1715–1718.
[31] W. Zhang, B. Paudel, L. Wang, J. Chen, H. Zhu, W. Zhang, Iteratively learning embeddings and rules for knowledge graph reasoning, International World Wide Web Conference(WWW'19) (2019).
[32] F. Zhang, N.J. Yuan, D. Lian, X. Xie, W.-Y. Ma, Collaborative knowledge base embedding for recommender systems, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 353–362.
[33] H. Ma, H. Yang, M.R. Lyu, I. King, Sorec: Social recommendation using probabilistic matrix factorization, in: Proceedings of the 17th ACM conference on Information and knowledge management, 2008, pp. 931–940.
[34] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, Wide & deep learning for recommender systems, in: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, 2016, pp. 7–10..
[35] H. Guo, R. Tang, Y. Ye, Z. Li, X. He, Deepfm: A factorization-machine based neural network for ctr prediction, in: Proceedings of the 26th International Joint Conference on Artificial Intelligence, 2017, pp. 1725–1731.
[36] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Proceedings of the 26th International Conference on Neural Information Processing Systems, 2013, pp. 3111–3119.
[37] X. Wang, X. He, Y. Cao, M. Liu, T. Chua, KGAT: knowledge graph attention network for recommendation, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4–8, 2019, 2019, pp. 950–958. doi:10.1145/3292500.3330989. URL:https://doi.org/10.1145/3292500.3330989..
[38] S. Rendle, Factorization machines with libfm, ACM Trans. Intell. Syst. Technol. 3 (3) (2012) 51–57.
[39] X. Yu, X. Ren, Y. Sun, Personalized entity recommendation: a heterogeneous information network approach, in: Proceedings of the 7th ACM International Conference on Web Search and Data Mining, 2014, pp. 283–292.
[40] R. Sun, Optimization for deep learning: theory and algorithms, CoRR abs/1912.08957. arXiv:1912.08957. URL:http://arxiv.org/abs/1912.08957..
[41] Y. Zhang, Explainable recommendation: Theory and applications, CoRR abs/1708.06409. arXiv:1708.06409. URL:http://arxiv.org/abs/1708.06409..