mathematics

MDPI

Article

Attention Knowledge Network Combining Explicit and Implicit Information

Shangju Deng [1,2], Jiwei Qin [1,2,*], Xiaole Wang [1,2] and Ruijin Wang [3]

[1] School of Information Science and Engineering, Xinjiang University, Urumqi 830046, China
[2] Key Laboratory of Signal Detection and Processing, Xinjiang Uygur Autonomous Region, Xinjiang University, Urumqi 830046, China
[3] School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China
* Correspondence: jwqin@xju.edu.cn

Abstract: The existing knowledge graph embedding (KGE) method has achieved good performance in recommendation systems. However, the relevancy degree among entities reduces gradually along the spread in the knowledge graph. Focusing on the explicit and implicit relationships among entities, this paper proposes an attention knowledge network combining explicit and implicit information (AKNEI) to effectively capture and exactly describe the correlation between entities in the knowledge graph. First, we design an information-sharing layer (ISL) to realize information sharing between projects and entities through implicit interaction. We innovatively propose a cross-feature fusion module to extract high-order feature information in the model. At the same time, this paper uses the attention mechanism to solve the problem of the decline of information relevance in the process of knowledge graph propagation. Finally, the features of KGE and cross feature fusion module are integrated into the end-to-end learning framework, the item information in the recommendation task and the knowledge graph entity information are interacted implicitly and explicitly, and the characteristics between them are automatically learned. We performed extensive experiments on multiple public datasets that include movies, music, and books. According to the experimental results, our model has a great improvement in performance compared with the latest baseline.

Keywords: recommendation system; knowledge graph embedding; multi-task learning; attention mechanism

MSC: 68T07

Citation: Deng, S.; Qin, J.; Wang, X.; Wang, R. Attention Knowledge Network Combining Explicit and Implicit Information. *Mathematics* **2023**, *11*, 724. https://doi.org/10.3390/math11030724

Academic Editors: Heui Seok Lim, Sanghyuk Lee, Yeongwook Yang and Imatitikua Aiyanyo

Received: 28 December 2022
Revised: 19 January 2023
Accepted: 21 January 2023
Published: 1 February 2023

1. Introduction

Currently, the amount of data has exploded, and recommendation systems have become one of the methods for solving data overload. Collaborative filtering (CF) [1] is widely used in various recommendation scenarios and is one of the most classic recommendation algorithms. As the most popular recommendation technique, CF utilizes the users' historical interaction information and the same preferences to provide the users with personalized recommendations. However, the CF method has the problems of cold start and data sparseness. To alleviate these problems and improve recommendation performance, researchers usually use the following: rich scene-related information, such as social networks [2], where the addition of social information not only reduces the sparsity of data, but also more accurately expresses users' preferences; user–item attribute [3], which provides richer types of information, allowing users to select more specific and accurate items and enrich the amount of data; and context information [4]. Although the addition of scene information improves recommendation performance to a certain extent, it ignores the correlation between the information, and such improvements are limited.

In further studies [5–9], researchers noted that information is interconnected, and through these associations, entity information (users or items) can be combined to form a knowledge graph (KG). For example, the LBSN [5] network is an integration of an RNN-based network and key–value memory network (KV-MN), and the model also uses the correlation information of the knowledge base to enhance the semantic representation of the model. Normally, KG is a graph with specific directions composed of multiple nodes and connections between nodes. The nodes in the graph represent entities (which can be users or items), and the connections between entities are called edges that represent the relationship between two entities. Profiting from the KG construction method, the algorithm based on KG has the advantages of interpretability and strong scalability.

Therefore, the KG algorithm with the above advantages has been well applied in recommendation systems. In the related research on using KGs to improve recommendation performance, a method based on knowledge graph embedding (KGE) [6,9,10] is one of them. For example, a deep knowledge perception network (DKN) [7] is a model that combines entity embedding and a convolutional neural network (CNN) [11]. The network integrates knowledge graph representation for news recommendation, which is a content-based click-prediction framework. The key part of the model is a perceptive convolutional neural network using multi-channel word entity alignment, which can dynamically aggregate users' history records and candidate news. Embedded-based methods have a high degree of flexibility, but they are more suitable for tasks such as application and link prediction than for personalized recommendations. Based on the KG path method, KG is usually regarded as a network of heterogeneous information, such as the personalized entity recommendation (PER) [8] proposed by Xiao Yu et al. The model uses the potential relationships in the KG original path to make different types of combinations, and the hidden information of the user's historical interaction is represented by the meta-path. However, the graph of the PER model relies on manual design, making it impossible to automate feature learning. RippleNet [6] utilizes user preferences to propagate in the KG, the model divides KG into multiple levels, including the preferences of users at different levels so as to tap the potential interests of users. However, because users have different preferences for different layers of KG, RippleNet cannot accurately capture user interests in each layer. The above models also have a common problem: how to obtain the potential feature information between the entity object in the knowledge graph and the item that needs to be predicted. Regarding the limitations of existing problems, this paper proposes a deep network framework (AKNEI) based on multiattention mechanism joint graph embedding, which uses KGE to assist in recommendation tasks. Recommendation tasks and KGE tasks are highly related to each other. The ISL layer embeds the KG entity formed by the user interaction history with the project to supplement and share information. The high-order features extracted by the multi-layer cross feature fusion network are fused with the information extracted by the ripple network of the attention mechanism to form an end-to-end model. This article makes the following contributions:

(1) We designed an ISL layer between KGE and recommendation tasks to connect KGE and recommendation tasks for feature sharing, automatically transfer interactive information during training, and obtain implicit semantics of entities and items. The ISL layer improves the model's anti-noise and generalization capabilities.
(2) We propose a cross feature fusion network that can explicitly extract features. This method can perform high-level explicit cross fusion of features, and the interaction of features occurs at the vector level, which reduces the number of parameters compared with the traditional neural network model that occurs at the element level. The network can retain the key information of each layer during the propagation process, preventing key information from being lost during the propagation process.
(3) Based on the ISL layer, we designed a multilayered corrugated network based on multihead attention, which can consider the changes in users' interests at different levels and achieve more accurate recommendations. Finally, combined with the cross feature fusion layer for high-order feature interaction, the shared information between

KG entities and items can be fully utilized. We conducted experiments on multiple datasets, and good results were achieved on both large-scale and small datasets.

2. AKNEI Model

In this part, the proposed AKNEI model is introduced, and we give a detailed explanation of our research methods and details.

2.1. Problem Statement

In the recommendation model combined with the KG graph, given a set $U = \{u_1, u_2, \cdots u_n\}$ with n users and a set $V = \{v_1, v_2, \cdots v_n\}$ with n items. The interactive information between the user and the item constitutes a matrix $Y = \{y_{wv} \mid u \in U, v \in V\}$. When $y_{uv} = 1$, it indicates that there have been historical interactions between users and items, such as watching, collecting and playing. Conversely, $y_{uv} = 0$ means that the user has never interacted with the item. The knowledge graph G existing in the model is connected by entity relationships to form $G = \{h, r, t\}$, where $h \in E$ is the head of the graph, $t \in E$ is the tail of the graph, $E = \{e_1, e_2, \cdots\}$ is the entity set of the knowledge graph, and $r \in R$ and $R = \{r_1, r_2, \cdots\}$ are the relationships between entities. For example, Mark Osborne, the director of the movie *Kung Fu Panda*, also made the movie *The Little Prince*. The two movies can be regarded as entities in a KG, and the common director can be regarded as the relationship between entities. Our goal is to use the interaction matrix Y and the knowledge graph G to analyze the user's behavior and predict the items V that the user is interested in, and these are items that users may like but with which they have never had historical interactions. The prediction function is $\hat{y}_{uv} = F(u, v; \theta, Y, G)$, and $\hat{y}_{uv}$ is the possible interaction probability between user u and item v, where θ is the parameter of the prediction function F.

2.2. Model Framework

The overall framework of our proposed AKNEI is shown in Figure 1. The input of the recommendation layer is user and item embedding, and the input of the KG layer is a triple graph G. The recommendation module is a set $\{(h, r, t)_1, \cdots (h, r, t)_n\}^k (k = 1, 2, \cdots n)$ of ripple graphs composed of user click history, where k is a ternary set of knowledge graphs of each layer of ripples. These collections interact with item embeddings, and the attention module is used to extract each layer of interactive information to form embeddings. CIN conducts high-level interactions with the item embeddings on the user preferences of each layer of the ripple network, forms the final embedding with the information extracted by the attention, and finally makes predictions. The ISL layer establishes a low-level information-sharing channel between the recommendation task and KGE, which can automatically learn feature interaction and complement information.

2.3. Information-Sharing Layer (ISL)

The ISL layer can implicitly interact the item information in the recommendation task with the entity information in the KG, and it can also supplement the two parts of the information. The design comes from [12]. As shown in Figure 2, for entity e in item v and the KG, its features $v_l \in \mathbb{R}^d$ and $e_l \in \mathbb{R}^d$ where d is the hidden layer dimension, their interaction process is

$$S_l = v_l e_l^T = \begin{bmatrix} v_l^1 e_l^1 & \cdots & v_l^1 e_l^d \\ \vdots & & \vdots \\ v_l^d e_l^1 & \cdots & v_l^d e_l^d \end{bmatrix} \tag{1}$$

$$S_l^T = e_l v_l^T = \begin{bmatrix} e_l^1 v_l^1 & \cdots & e_l^1 v_l^d \\ \vdots & & \vdots \\ e_l^d v_l^1 & \cdots & e_l^d v_l^d \end{bmatrix} \tag{2}$$

Among them, $S_l \in \mathbb{R}^{d \times d}$. Through this interaction process, the information between the item and the entity is shared, and the feature interaction is displayed and modeled. Using this matrix as the input of the next layer of interaction, the process is as follows:

$$v_{l+1} = \left(w_l^{VV} S_l + w_l^{EV} S_l^T + b_l^V\right) + v_1 + e_1 \tag{3}$$

$$e_{l+1} = \left(w_l^{VE} S_l + w_l^{EE} S_l^T + b_l^E\right) + v_1 + e_1 \tag{4}$$

where $w_l \in \mathbb{R}^d$ and $b_l \in \mathbb{R}^d$ are trainable weights and bias terms, respectively. Through this operation, the interactive matrix space $\mathbb{R}^{d \times d}$ is compressed into the vector space $\mathbb{R}^d$. The ISL layer usually has a better information interaction effect at the lower layer of the network. The deeper the network layer, the more special the higher-order features, and the transferability is significantly reduced [13].

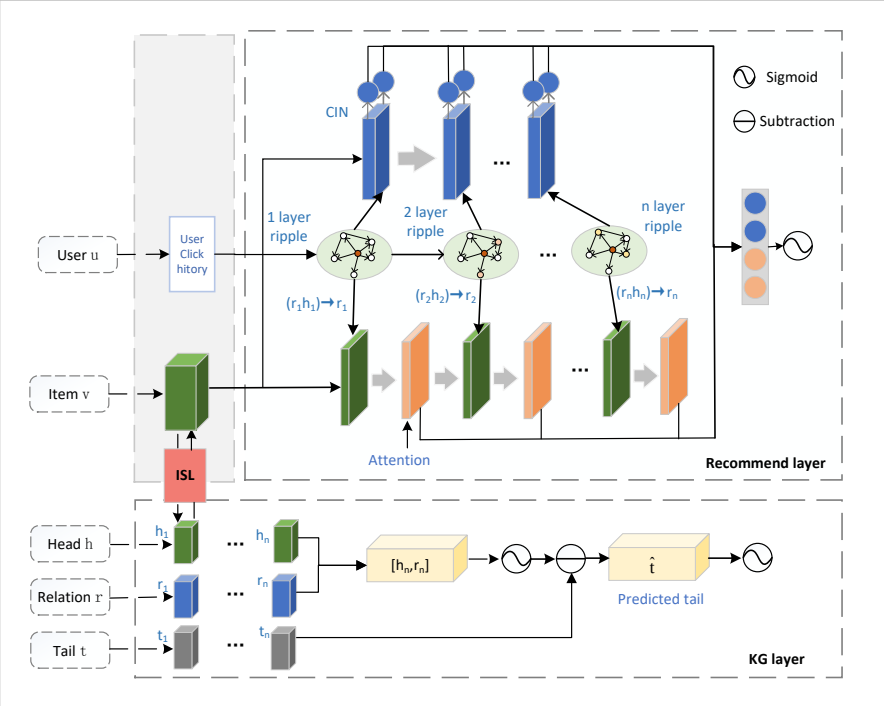

Figure 1. AKNEI framework diagram, composed of a KG layer, cross feature fusion network layer and a multilayer corrugated network with attention. Through the ISL layer, an information-sharing channel is established between the KG and the recommendation task, and information sharing is realized at the bottom layer.

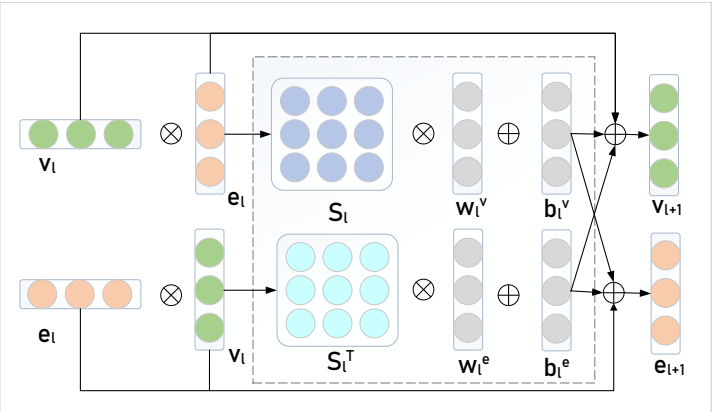

Figure 2. Information sharing at the ISL layer.

ISL Analysis

We next introduce the models related to the ISL layer theory to prove the feature interaction capabilities of ISL and explain them conceptually.

Factorization machine (FM) The factorization machine [14] is a commonly used method in recommendation systems. FM uses factorization to interactively model the input features and has a good estimate of the sparse problem. The equation of the 2-degree factorization machine model is

$$\hat{y}(x) = y_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d} \sum_{j=i+1}^{d} \langle p_i, p_j \rangle x_i x_j \tag{5}$$

Denote the i-th value of the input vector x as x_i, p is the weight of the input vector, and $\langle \cdot, \cdot \rangle$ represents the dot product calculation between vectors. The above formula is similar to the 1-layer ISL. The L1-norm formula for the interaction of $\mathcal{V}_1$ and e_1 is as follows:

$$\begin{aligned} \|v_1\|_1 &= \left| \sum_{i=1}^{d} \sum_{j=1}^{d} \langle w_i, w_j \rangle v_i e_j \right| \\ \|e_1\|_1 &= \left| \sum_{i=1}^{d} \sum_{j=1}^{d} \langle w_i, w_j \rangle e_i v_j \right| \end{aligned} \tag{6}$$

where $\langle w_i, w_j \rangle = w_i + w_j$ is the sum of scalars; below, we provide the proof of the above formula. Without retaining the previous layer of information, the formula is written as follows:

When $l = 1$

$$\begin{aligned} v_1 &= v e^T w_0^{VV} + e v^T w_0^{EV} + b_0^V \\ &= \left[v_1 \sum_{i=1}^{d} w_0^{VV(i)} e_i \cdots v_d \sum_{i=1}^{d} w_0^{VV(i)} e_i \right]^T \\ &+ \left[e_1 \sum_{i=1}^{d} w_0^{EV(i)} v_i \cdots e_d \sum_{i=1}^{d} w_0^{EV(i)} v_i \right] \\ &+ \left[b_0^{V(0)} \cdots b_0^{V(d)} \right]^T \end{aligned} \tag{7}$$

The L1-norm of V is

$$\begin{aligned} \|v_1\|_1 &= \left| \sum_{j=1}^{d} v_j \sum_{i=1}^{d} w_0^{VV(i)} e_i + \sum_{j=1}^{d} e_j \sum_{i=1}^{d} w_0^{EV(i)} v_i \right. \\ &\left. + \sum_{i=1}^{d} b_0^{V(d)} \right| \\ &= \left| \sum_{i=1}^{d} \sum_{j=1}^{d} \left(w_0^{EV(i)} + w_0^{VV(i)} \right) v_i e_j + \sum_{i=1}^{d} b_0^{V(d)} \right| \end{aligned} \tag{8}$$

The proof for e_1 is similar.

Different from FM, the FM parameter is the dot product of the interaction vector weight parameters, while ISL is the sum of the weight parameters. Compared to FM, the total amount of parameters decreases.

Cross-stitch network Cross-stitch network [12] units can establish sharing between two tasks, and at the same time, carry out a specific representation of the two tasks:

$$\begin{bmatrix} \hat{y}_N^{ij} \\ \hat{y}_M^{j} \end{bmatrix} = \begin{bmatrix} \delta_{NN} & \delta_{NM} \\ \delta_{MN} & \delta_{MM} \end{bmatrix} \begin{bmatrix} y_N^{ij} \\ y_M^{ij} \end{bmatrix} \tag{9}$$

where y_n and y_m are given two task maps, $delta$ is the weight between the two task graphs, and i and j are shown at position (i, j) in the figure. Input is provided for the next layer of filters through linear combinations. This is similar to our ISL unit. Without considering the bias, we can write the ISL in the following form:

$$\begin{bmatrix} v_{l+1} \\ e_{l+1} \end{bmatrix} = \begin{bmatrix} w_l^{VV} e_l^T & w_l^{EV} v_l^T \\ w_l^{VE} e_l^T & w_l^{EE} v_l^T \end{bmatrix} \begin{bmatrix} v_l \\ e_l \end{bmatrix} \tag{10}$$

Similar to cross-stitch networks, the ISL unit can adapt to specific tasks through weight distribution and obtain shared information between different tasks.

2.4. KG Layer

The knowledge graph module expresses the node and the association between nodes in the KG graph in the form of vectors. There are many methods based on KGE, such as the relationship embedding of Antoine Bordes et al. [15], and the entity relationship embedding model used by Yankai Lin and others for knowledge graph completion [16], Hanxiao Liu et al.'s multirelation embedding model [17] and Maximilian Nickel's holographic embedding model of knowledge graphs [18]. Their models have brought us different knowledge graph embedding methods. Our model uses the characteristics of the head node h and uses the relationship r between the nodes in the knowledge graph to predict the tail node t. The operation is as follows:

$$O(x) = \sigma(Wx + b) \tag{11}$$

$$O^l(x) = O(O(\cdots O(x)) \tag{12}$$

$$h_l = \mathbb{N}_{v \sim S(h)}\{I(v, h)\} \tag{13}$$

$$\hat{t}_l = O^l([h_l, r_l]) \tag{14}$$

where $\sigma(\bullet)$ is a nonlinear activation function and $S(h)$ is the set of all nodes h in KG, $\mathbb{N}$ is the set of entities associated with h. Here, we briefly describe the interaction process through the ISL layer as $I(v, h)$, and function f_{KG} is the score function:

$$\text{score}(h, r, t) = f_{KG}(t, \hat{t}) = \sigma(t \cdot \hat{t}) \tag{15}$$

2.5. Attention Ripple Layer

This layer belongs to the recommendation module in the AKNEI framework, which uses the dissemination of user preferences on the KG and is an improvement on the framework proposed by RippleNet [6]. The principle of corrugated layer propagation is to start from a clicked entity and extend outward along the different relationships between entities. As the process of outward propagation, the correlation between entities gradually declines, similar to the formation of water droplets on the water surface to form outward diffusion corrugation; the smaller the outer layer, the smaller the corrugation, as shown in Figure 3. The KG is used to spread user preferences in layers [6]. Considering that the preferences of physical users in each hop (each layer) are different, we add a multihead attention mechanism [19] to assign different weights to each layer. This makes the model more accurate in predicting user preferences. The multihead attention mechanism has excellent performance in modeling complex relationships. For example, machine translation [20] and sentence embedding [21] have shown excellent performance, and they have also been applied in the similarity capture of graph embedding nodes [22]. We are given an interaction matrix $Y = I(v, h)$ and a knowledge graph G. The user's entity composition set at the k layer is as follows:

$$\xi_u^k = \left\{ h \mid (h, r, t) \in G, h \in \xi_u^{k-1} \right\}, k = 1, 2, 3, \ldots, N \tag{16}$$

where $\xi_u^0 = V_u = \{v \mid y_{uv} = 1\}$ is the collection of items that the user has interacted with, which can be regarded as the 0th layer in the ripple. The k-th jump obtains the ripple set as ξ_u^k, which is derived from the upper layer set ξ_u^{k-1} in the ripple diagram:

$$Q_u^k = \left\{ g(h,r,t) \mid (h,r,t) \in G, h,t \in \xi_u^{k-1} \right\}, k = 1,2,3,\ldots,N \tag{17}$$

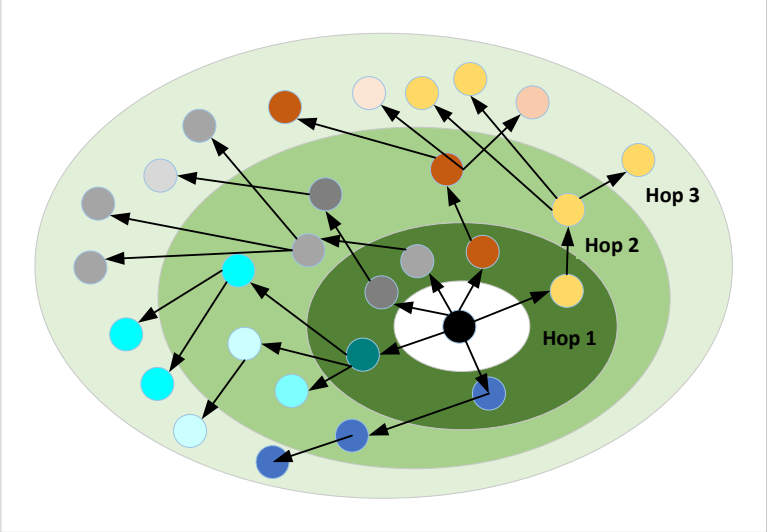

Figure 3. Ripple propagation diagram. Dots are entities, colors represent different categories, and arrows are the relationships between entities. As the number of hops between entities increases, the correlation decreases.

As shown in Figure 1, item v comes from item embedding $v \in \mathbb{R}^d$, and the item v is merged with the head node h_i and the relationship r_i in each triplet (h_i, r_i, t_i) in the ripplenet network ξ_u^k to allocate the association probability:

$$p_i = \text{softmax}\left(v^T \mathrm{R}_i \, \mathrm{h}_i\right) = \frac{\exp\left(v^T R_i h_i\right)}{\sum_{(h,r,t)\in S_u^1} \exp\left(v^T R h\right)} \tag{18}$$

here $R_i \in \mathbb{R}^{d\times d}$ is the embedding of relation r_i, $h_i \in \mathbb{R}^d$ is the embedding of h_i, and p_i is the similarity between item v and head node h in relation space R_i. Probabilistically weight the similarity with the tail node t in Q_u^1:

$$b_u^1 = \sum_{(h_i,r_i,t_i)\in Q_u^1} p_i t_i \tag{19}$$

where $t_i \in \mathbb{R}^d$ is the embedding of the tail node t_i, and the vector b_u^1 is the correspondence of user's u click history v_u to item v. Performing the above operations for each layer of the corrugated network can obtain the second and nth layer responses b_u^2 and b_u^n. The difference from RippleNet is that we input the corresponding input of each layer to the multihead attention module instead of simply adding them. The correlation between any two layers b_u^k and b_u^{k+1} in the above is defined under the attention head i. The specific operations are as follows:

$$\alpha_{k,k+1}^{(i)} = \frac{\exp\left(\psi^{(h)}\left(b_u^k, b_u^{k+1}\right)\right)}{\sum_{k=1}^{M} \exp\left(\psi^{(h)}\left(b_u^m, b_u^1\right)\right)} \tag{20}$$

$$\psi^{(i)}\left(b_u^k, b_u^{k+1}\right) = \left\langle W_{\text{Value}}^{(i)} \, b_u^k, W_{\text{Key}}^{(i)} \, b_u^{k+1} \right\rangle \tag{21}$$

where $\psi^{(i)}(\cdot,\cdot)$ is a similarity function, which can be a neural network or an inner product. In the model, we use the inner product because it is simple and efficient. $W^{(i)}_{\text{Value}}, W^{(i)}_{\text{Key}} \in \mathbb{R}^{d' \times d}$ the transformation matrix that maps the original space $\mathbb{R}^d$ to the new feature space $\mathbb{R}^{d'}$. Next, we update the b_u^{k+1} feature under the i space of the attention head by combining the coefficient $\alpha^{(i)}_{k,k+1}$ of all relevant features:

$$\tilde{b}^{k+1}_{u^{(i)}} = \sum_{k=1}^{M} \alpha^{(i)}_{k,k+1} \left(W^{(i)}_{\text{Value}}\, b_u^{k+1} \right) \tag{22}$$

where $\tilde{b}^{k+1}_{u^{(i)}}$ is the feature updated in i space, $W^{(i)}_{\text{Value}} \in \mathbb{R}^{d' \times d}$.

For the situation where there are multiple combinations of multiple features, multiple heads are used to create different head spaces for different feature interactions. The combined features in all subspaces are as follows:

$$\tilde{b}_u^{k+1} = \tilde{b}^{k+1}_{u^1} \oplus \tilde{b}^{k+1}_{u^2} \oplus \cdots \oplus \tilde{b}^{k+1}_{u^i} \tag{23}$$

where $\oplus$ is a connection symbol that connects multiple vectors into one vector and i is the input head in the multihead attention mechanism. Finally, we retain the original combined features and obtain

$$b^{k+1}_{u^{\text{Res}}} = \text{ReLU}\left(\tilde{b}_u^{k+1} + W_{\text{Res}} b_u^{k+1} \right) \tag{24}$$

$$\hat{y} = \sigma\left(w^T \left(\tilde{b}^1_{u^{\text{Res}}} \oplus \tilde{b}^2_{u^{\text{Res}}} \oplus \cdots \oplus \tilde{b}^k_{u^{\text{Res}}} \right) + b_{\text{bias}} \right) \tag{25}$$

where $W_{\text{Res}} \in \mathbb{R}^{d'\text{i} \times d}$ is the item matrix in the case of dimensional mismatch [23], w^T is the weight matrix, b_{bias} is the amount of paranoia, and $\sigma = 1/(1+e^{-x})$ converts the value to the user's click-through rate. We can stack multiple such interactive layers for feature update, and the input of the current network layer is the output of the previous network layer so that multilayer features can be modeled. The multihead attention block diagram is shown in Figure 4. For an explanation, see [24].

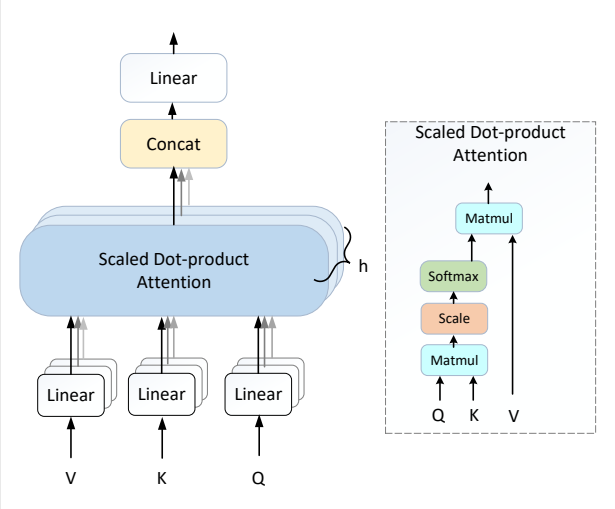

Figure 4. Block diagram of the multihead attention mechanism, where MATMUL is the matrix product, scale is the matrix of different dimensions, softmax is the activation function, and h is the number of heads.

2.6. Cross Feature Fusion Network (CFFN)

At present, traditional neural DNNs [11], CNNs [25] and other networks are used to automatically learn feature interaction models, most of which are based on the framework of factorization machines [14,26,27], and multilayer fully connected neural networks are

used for automatic feature learning. For high-level interactions, however, the implicit interaction process is unexplainable and unknown. Feature interaction is at the element level, and the amount of parameters is obviously more than that at the feature level, which does not conform to the original idea of the factorization machine. The cross-feature fusion module performs explicit feature interaction and has good interpretability. The feature is interaction at the vector level, and the amount of parameters is relatively small [28]. In the process of propagation, the network can retain the key information of each propagation layer and finally aggregate it so as to prevent the loss of key feature information during the propagation process. The cross feature fusion module is shown in Figure 5. The specific operations are as follows:

$$X^1 = W^{m\times d} b_u^1 \tag{26}$$

$$X_{h,*}^k = \sum_{i=1}^{H_{k-1}} \sum_{j=1}^{m} W_{ij}^{k,h} \left(X_{i,*}^{k-1} \circ X_{j,*}^n \right) \tag{27}$$

where $W^{m\times d}$ is a mapping matrix. The product operation of vector b_u^1 and matrix $W^{m\times d}$ can map vector b_u^1 to $m \times d$-dimensional space, $X_{j,*}^1$ is the j-th column vector of the embedding matrix X^1, $X_{j,*}^n$ is the embedding vector $1 \leq n < k$ of the ripple network corresponding to the CFFN propagation layer, $X^k \in \mathbb{R}^{H_k \times d}$ is the kth layer output matrix in the cross feature fusion network, and the number of feature vectors is expressed as H_k. Set the first layer to $H_0 = m$, $1 \leq h \leq H_k$, and $W_{ij}^{k,h} \in \mathbb{R}^{H_{k-1}\times m}$ to be the parameter matrix of the h-th eigenvector. $\circ$ represents the Hadamard product, for example, $(a_1, a_2) \circ (b_1, b_2) = (a_1 b_1, a_2 b_2)$. The number of cross feature fusion layers determines the degree of high-order feature interaction. The pooling operation connects the output layer and the hidden layer, so the output of each layer depends on the previous hidden layer and additional layers. Therefore, it can be ensured that the output unit can obtain characteristic interaction modes of different orders. The above formula is closely related to the CNN, as shown in Figure 5. The intermediate tensor Z^{k+1} is introduced, which is the outer product of the hidden layer X^{k-1} and the n-th ripple propagation embedding matrix X^n. This tensor is similar to the graph in CNN, the embedding dimension d can be seen as the number of channels, and $W^{k,h}$ is a filter that extracts the features of the d layer to obtain the hidden vector $X_{i,*}^{k+1}$. Figure 1 shows the propagation process of the cross feature fusion network in the model. We set the maximum number of layers as T and then $k \in [1, T]$ and connect all the layers as follows:

$$\varphi_i^k = \sum_{j=1}^{d} X_{i,j}^k \tag{28}$$

where $i \in [1, H_k]$, the pooling vector of length H_k in the kth layer is $\varphi^k = \left[\varphi_1^k, \varphi_2^k, \cdots, \varphi_{H_k}^k\right]$, and the connection between the hidden layer and the output layer is expressed as

$$\varphi^+ = \frac{1}{2}\left[\varphi^1, \varphi^2, \cdots, \varphi^T\right] \in \mathbb{R}^{\sum_{i=1}^{T} H_i} \tag{29}$$

The final prediction result is

$$\hat{y} = \frac{1}{1 + \exp(\varphi^+ w^o)} \tag{30}$$

where w^o is the regression parameter.

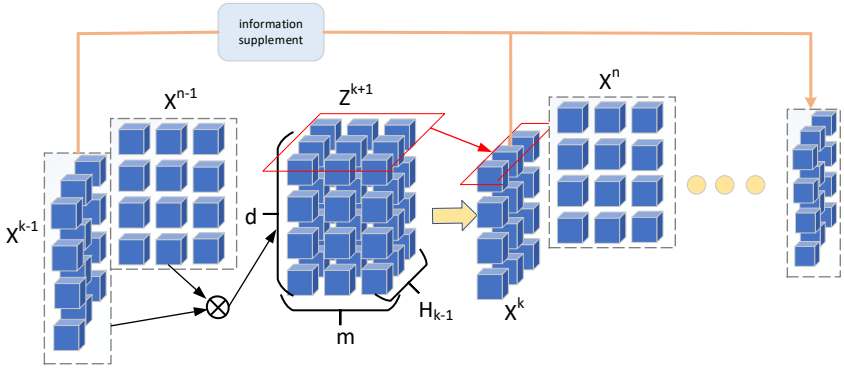

Figure 5. Calculating the outer product of X^{k-1} and X^n to obtain the intermediate tensor Z^{k+1}, the red mark indicates that the intermediate tensor is compressed into an embedding vector.

Cross Feature Fusion Network Analysis

In this part, we discussed the interactive characteristics of the cross feature fusion network. We first set the number of features of the hidden layer and the number of fields m to be equal. The first layer contains h feature maps, and its formula can be written as

$$x_h^1 = \sum_{\substack{i\in[m]\\ j\in[m]}} W_{i,j}^{1,h}\left(x_i^0 \circ x_j^0\right) \tag{31}$$

where $[m]$ is a positive integer less than m. It can be seen in the formula that each feature model of the first layer has a paired interaction with $O(m^2)$ coefficients. In the same way, the second layer is calculated as follows:

$$\begin{aligned} x_h^2 &= \sum_{\substack{i\in[m]\\ j\in[m]}} W_{i,j}^{2,h}\left(x_i^1 \circ x_j^1\right) \\ &= \sum_{\substack{i\in[m]\\ j\in[m]}} \sum_{\substack{l\in[m]\\ k\in[m]}} W_{i,j}^{2,h} W_{l,k}^{1,i}\left(x_i^0 \circ x_k^0 \circ x_l^0\right) \end{aligned} \tag{32}$$

Our purpose is to prove that the number of cross feature fusion network parameters is only $O(\mathrm{km}^3)$, while the parameters of the classical k-order polynomial are $O(\mathrm{m}^3)$. The feature mapping of the K layer can be summarized as

$$\begin{aligned} x_h^k &= \sum_{\substack{i\in[m]\\ j\in[m]}} W_{i,j}^{k,h}\left(x_i^{k-1} \circ x_j^0\right) \\ &= \sum_{\substack{i\in[m]\\ j\in[m]}} \cdots \sum_{\substack{r\in[m]l\in[m]\\ t\in[m]s\in[m]}} W_{i,j}^{k,h} \cdots W_{l,s}^{1,i} \underbrace{\left(x_j^0 \circ \cdots \circ x_s^0 \circ x_l^0\right)}_{k \text{ vectors}} \end{aligned} \tag{33}$$

Let $\beta = [\beta_1, \ldots, \beta_m] \in \mathbb{N}^d$ denote multiple index information, then $|\beta| = \sum_{i=1}^m \beta_i$. Because the 0th layer is used to represent the final feature map, we replace x_i^0 with x_i for convenience. Superscripts are used to denote operations, such as $x_i^3 = x_i \circ x_i \circ x_i$. Let $F\varphi_k(X)$ be a polynomial of degree k containing multiple vectors:

$$F\varphi_k(X) = \left\{ \sum_{\beta} w_\beta x_1^{\beta_1} \circ x_2^{\beta_2} \circ \ldots \circ x_m^{\beta_m} | 2 \le |\beta| \le k | \right\} \tag{34}$$

The parameter of the above classical vector polynomial is $O\left(m^k\right)$, and the cross feature fusion network parameter formula is as follows:

$$\hat{w}_\beta = \sum_{i=1}^{m} \sum_{j=1}^{m} \sum_{Q \in P_\beta} \prod_{t=2}^{|\beta|} W_{i,Q_t}^{t,j} \tag{35}$$

where $Q = \left[Q_1, Q_2, \cdots, Q_{|\beta|}\right]$ is a multi-index and $P_{|\beta|}$ is the permutation set of all indices.

2.7. Learning Algorithm

We combine each module to form the complete loss function as

$$\begin{aligned} \mathcal{L} =& \mathcal{L}_{CIN} + \mathcal{L}_{Ripple} + \mathcal{L}_{KG} \\ =& \sum_{u \in U, v \in V} \mathcal{J}(\hat{y}_{uv}, y_{uv}) + \frac{1}{2} \sum_{r \in R} \left\| \lambda_1 I_r - \lambda_1 E^T R E \right\|_2^2 \\ &+ \frac{1}{2} \left(\lambda_2 \|V\|_2^2 + \lambda_2 \|E\|_2^2 + \lambda_2 \sum_{r \in R} \|R\|_2^2 \right) \end{aligned} \tag{36}$$

where V represents the item embedding matrix, E represents the entity embedding matrix, I_r is the slice of the embedding tensor I in the KG representing the relationship r, and the relevant information r in KG is embedded in the matrix to obtain R. In the first term, $\mathcal{J}(\cdot,$, is the loss function between the calculated real data value $\hat{y}_{uv}$ and the model prediction value $\hat{y}_{uv}$, called cross entropy loss. The second term is the square error between the true value in the KG and the reconstruction matrix. The last item is a regularization term to prevent overfitting.

The optimization algorithm we choose is stochastic gradient descent (SGD). To improve computational efficiency, ref. [29] provided us with a negative sampling strategy, which can improve sampling efficiency. We apply the parameter analysis of the specific model in the experimental part.

3. Links to Existing Work

In this section, we introduce the difference between related work and our method. At present, deep learning is increasingly widely used in recommendation systems, and it has good performance in many recommendation scenarios to fit the interaction behavior of users and items into the neural network model [11]. For example, neural collaborative filtering [30] is used for the high-level interaction of users and items. Our method differs from the above methods in terms of high-level feature interaction. The cross feature fusion network is different from traditional neural networks. The displayed feature interaction has good interpretability, and its feature interaction effectively reduces the number of parameters at the vector level.

The interpretability in the recommendation system refers to the reason why the user likes a certain item. By analyzing the reason, a more accurate personalized recommendation result can be put forward for the user. At present, interpretability is generally based on community tags [31], social networks [32], and emotional semantics [33,34]. In the recommendation module, RippleNet is based on a KG to find user interests. It has strong interpretability by tracking user history and the path of related items [6]. Our feature interaction layer uses a feature matrix method similar to MKR [35] and RuleRec [36] to achieve the complementarity of KG information and recommended information. The existence of the ISL layer makes our model share implicit information. Our method can be regarded as a multitask framework, using a KG for auxiliary recommendation, which can automatically transfer information and learn feature interactions.

4. Experiment

4.1. Public Datasets

- **MovieLens-1M** (https://grouplens.org/datasets/movielens/1m/ accessed on 1 December 2022): It is a widely used movie recommendation dataset. It includes approximately 1 million ratings from 6040 users on 3706 movies, with ratings ranging from 1 to 5.
- **MovieLens-20M** (https://grouplens.org/datasets/movielens/20m/ accessed on 1 December 2022): It is a dataset belonging to the MovieLens website, which contains the rating data of approximately 20 million users, with rating values ranging from 1 to 5.
- **Book-Crossing** (http://www2.informatik.uni-freiburg.de/cziegler/BX/ accessed on 1 December 2022): It is a benchmark dataset for book recommendation, which contains approximately 1.1 million ratings of 270,000 books by 90,000 users, with ratings ranging from 1 to 10.
- **Last.FM** (https://grouplens.org/datasets/hetrec-2011/ accessed on 1 December 2022): It is a dataset of an online music website that contains the listening information of 2000 users.

We need to preprocess the above datasets before the experiment. For the MovieLens-1M and MovieLens-20M datasets, we use a score greater than or equal to 4 as positive feedback. Book-Crossing and Last.FM consider sparsity for the question; listening information and scoring information are regarded as positive feedback. Microsoft Satori is used as the KG construction engine to extract knowledge triples. Detailed information about the dataset is shown in Table 1.

Table 1. Details of four public datasets.

	MovieLens-20M	MovieLens-1M	Book-Crossing	Last.FM
#user	138,159	6036	17,860	1872
#item	16,954	2445	14,967	3846
#interactions	13,501,622	753,772	139,746	42,364
#entities	102,569	182,011	77,903	9366
#relations	32	12	25	60
#KG triples	499,474	20,782	19,876	15,518
#sparsity level	0.9947	0.9489	0.9994	0.9941

4.2. Baseline Model

We compare the model proposed in this article with the following baseline model one by one:

- **LibFM [37]**: It combines general features with a near factorization model to make recommendations.
- **PER [8]**: A network that combines items and users and the interactive information between them.
- **Wide & Deep [11]**: A classic model combined with DNN neural network.
- **DKN [7]**: It is a news recommendation that uses multichannel and word-aligned methods to integrate CNNs.
- **RippleNet [6]**: The network forms a multi-layer knowledge graph network through the user's interaction history through similar category information, and captures user preference information at different levels.
- **MKR [35]**: It is a feature model that combines multiple tasks, in which the KG is used as an auxiliary for recommendation tasks.
- **KGCN [38]**: It uses the relationship attributes of the KG to mine the associations of items to learn the potential interests of users.
- **Ripp-MKR [39]**: This model is an end-to-end recommendation method incorporating knowledge graph embeddings, which enhances the representation of latent information.

- **CAKR [40]**: A new method is designed to optimize the feature interaction between items and corresponding entities in knowledge graphs, and a feature intersection unit combined with the attention mechanism is proposed to enhance the recommendation effect.

4.3. Experimental Setup

We set the corresponding hyperparameters for the four datasets. As shown in Table 2, other baseline hyperparameters default to their original settings. The parameter d in the table is the embedding dimension of the model, parameter H is the number of layers of the RippleNet, η represents the learning rate of the recommended task, μ is the weight of the KG, and λ represents the L2 regularization weight. In terms of evaluation indicators, we use AUC and ACC to evaluate our model. The AUC is an important indicator for evaluating the quality of a classifier. The higher the AUC and ACC values, the better. Divide the dataset at a ratio of 6:2:2.

Table 2. Dataset parameter setting.

Dataset	Parameters
MovieLens-20M	$d = 90$, $H = 3$, $\eta = 0.0005$, $\mu = 0.05$, $\lambda = 6 \times 10^{-5}$, $\theta = 0.05$
MovieLens-1M	$d = 90$, $H = 3$, $\eta = 0.0005$, $\mu = 0.05$, $\lambda = 6 \times 10^{-5}$, $\theta = 0.08$
Book-Crossing	$d = 120$, $H = 3$, $\eta = 4 \times 10^{-6}$, $\mu = 0.5$, $\lambda = 5 \times 10^{-5}$, $\theta = 0.09$
Last.FM	$d = 120$, $H = 3$, $\eta = 4 \times 10^{-3}$, $\mu = 0.09$, $\lambda = 8 \times 10^{-6}$, $\theta = 0.09$

4.4. Experimental Results

We performed click-through rate (CTR) predictions of different models on the 4 datasets. In the top-K recommendation, we choose Recall@ as the evaluation indicator of the model. We show the experimental results in Table 3 and Figure 6 in the form of tables and graphs respectively. The experimental analysis is as follows:

- Compared with other baselines, DKN has the worst performance. Whether it is on movie data with large data volumes or small and sparse music and book datasets, its performance is not satisfactory. DKN entity embedding before input cannot participate in the learning process.
- From the comparison experiment, PER performs better than DKN on each dataset but the performance is not good enough compared to other baselines. The reason is that the metapath defined by the PER has difficulty achieving the optimal value and cannot integrate heterogeneous relationship information.
- The performance of LibFM and Wide&Deep is relatively good on the four datasets. It shows that the model has a high information utilization rate for KG, and can extract relevant information well to improve recommendation performance.
- RippleNet has shown excellent performance, which shows that it has a significant effect in capturing user preferences. From the comparison of the four datasets, RippleNet has reduced performance in the case of sparse data and has a strong dependence on data density.
- Compared with the MKR and the KGCN, the KGCN is more sensitive to the sparsity of data and performs weaker on sparse book datasets, but the KGCN performs better than the MKR on datasets with large data volumes, indicating that the KGCN is more suitable for obtaining large data on recommended occasions.
- CAKR benefits from its improved interaction module, and the feature intersection unit of the attention mechanism has certain advantages in complementing information. Therefore, the model performs better on sparse datasets.
- It can be seen from the experimental results that the AKNEI model proposed in this paper performs better than the baseline model mentioned above on the four datasets. In the MovieLens-20M dataset, the AUC increases by 1.1–18.6%, and the ACC increases by 2.3–19.9%. AUC in MovieLens-1M increases by 1.6–29.3%, ACC increases by 1.9–30.7%, AUC increases by 0.5–16.1% in Book-Crossing, ACC increases

by 0.2–14.7%, Last.FM AUC increases by 3.1–26.6%, and ACC increases by 0.7–23.1%. This proves that our model has a higher utilization of user behavior data.

Table 3. The prediction result under the public dataset.

Model	Movielens-20M		Movielens-1M		Book-Crossing		Last.FM	
	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC
LibFM	0.9587	0.9055	0.8921	0.8122	0.6855	0.6402	0.7774	0.7098
PER	0.8341	0.7893	0.7055	0.6621	0.6238	0.5884	0.6321	0.5963
Wide&Deep	0.9635	0.9221	0.8988	0.8203	0.7123	0.6245	0.7564	0.6887
DKN	0.8012	0.7565	0.6551	0.5892	0.6123	0.5878	0.6001	0.5812
RippleNet	0.9675	0.9122	0.9115	0.8323	0.7254	0.6501	0.7658	0.6891
MKR	0.9702	0.9198	0.9121	0.8334	0.7297	0.6767	0.7923	0.7501
KGCN	0.9742	0.9223	0.9105	0.8312	0.6753	0.6221	0.7932	0.7203
Ripp-MKR	-	-	0.9220	0.8450	0.7400	**0.7120**	0.7990	0.7500
CAKR	-	-	0.9240	0.8490	**0.7460**	0.7070	0.8020	0.7545
AKNEI	**0.9851**	**0.9447**	**0.9271**	**0.8503**	0.7283	0.6887	**0.8177**	**0.7554**

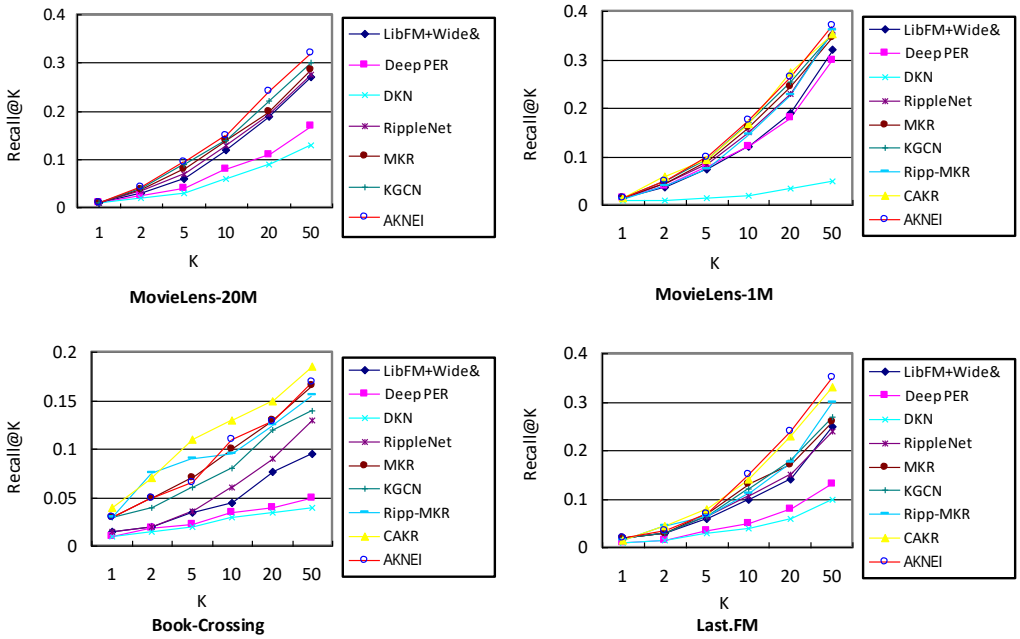

Figure 6. Recall@K results of multiple baseline models.

4.5. Influence of Different Modules

4.5.1. Impact of ISL Module, Cross Feature Fusion Network Module and RippleNet Module

In this part, we analyze the performance of each module in the datasets Movielens-1M and Last.FM. The results are shown in Figure 7. For the Last.FM dataset, the cross feature fusion network layer has the greatest impact on the model, indicating that the cross feature fusion network can alleviate the problem of data sparseness to a certain extent. For MovieLens-1M with more data, the performance of the ISL layer is better, indicating that the interaction between the item and the entity can obtain more valuable information. Through the comparison of the two datasets, the RippleNet and the ISL modules are more sensitive to the sparsity of the data, but they perform better when the amount of data is relatively abundant. The AKNEI model integrates the advantages of each module, which makes the model more generalized and able to achieve better performance in different recommended scenarios.

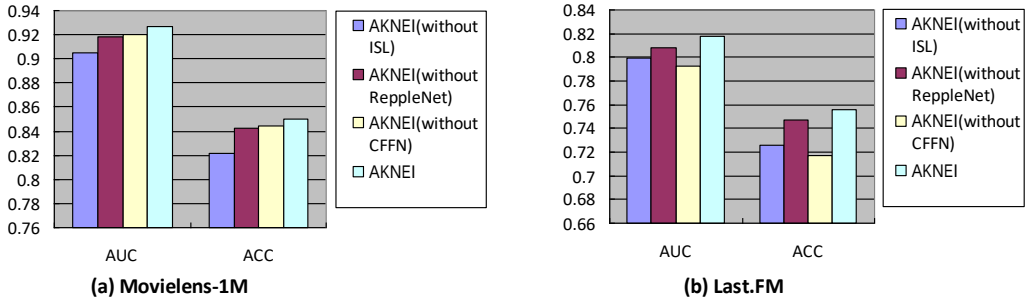

Figure 7. Influence of different modules.

4.5.2. Impact of RippleNet Layers and Attention Module

In this part, we analyze the number of layers of RippleNet and the influence of the attention module on model performance. As shown in Table 4, by changing RippleNet layer number H, we observe the performance changes. The model performance is best when H is 3. This phenomenon is attributed to the reality that the user's interest information will decrease as the graph propagates. At longer distances, the influence of noise will be greater than the effective information. When the distance is too close, it is difficult to dig out relevant information and dependencies between entities. Table 5 shows the influence of the attention module. It is not difficult to see that the addition of the attention module improves the performance of the model. This is because the attention module adds different weights to each layer of information to distinguish users' preferences for different layers.

Table 4. Influence of the number of Ripple layers.

H	1	2	3	4
Movielens-20M	0.9824	0.9835	**0.9851**	0.9843
Movielens-1M	0.9223	0.9254	**0.9271**	0.9246
Book-Crossing	0.7246	0.7279	**0.7283**	0.7280
Last.FM	0.7167	0.8173	**0.8177**	0.8169

Table 5. The influence of the attention module.

	AKNEI	AKNEI$_{w/o\ ATT}$
Movielens-20M	**0.9851**	0.9810
Movielens-1M	**0.9271**	0.9237
Book-Crossing	**0.7283**	0.7163
Last.FM	**0.8177**	0.8097

4.6. Parametric Analysis

In this part, we discuss the impact of embedding dimensions and KGE weights on model performance without changing other parameters.

4.6.1. Impact of Embedding Dimension

We conducted an experimental analysis on the relationship between the size of the embedding dimension and the expressive power of the model. The change of the model with the embedding dimension is shown in Figure 8; the performance of the model will increase as the embedding dimension increases, and will decrease after reaching the critical point. Because a larger dimensional embedding can increase the expressiveness of the model, but the model will fit more noise, too-high dimensional embedding will cause the model to appear as over-fitting.

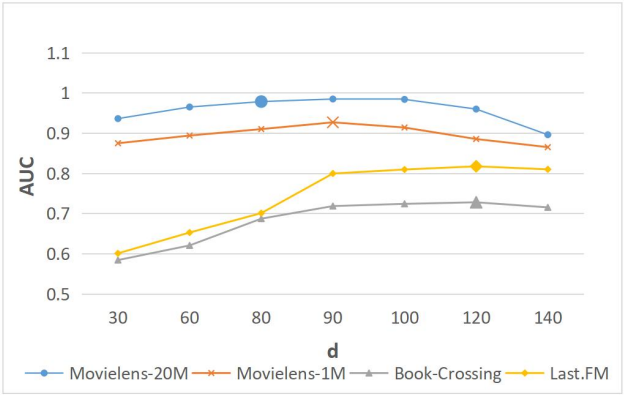

Figure 8. Dimension of embedding.

4.6.2. Impact of KEG Weight

As shown in Figure 9, the performance of the model shows a trend of first rising and then falling as the KGE weight increases. This is because a small weight will make the regularization constraint of the KGE term too small and cause overfitting of the model, while too large a weight will cause the objective function to be misled by KGE and deviate from the true value.

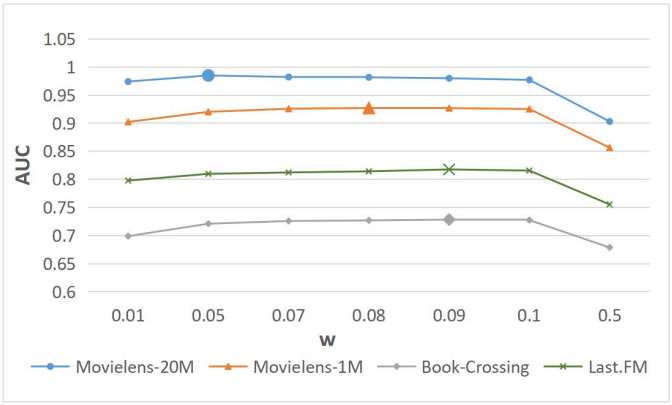

Figure 9. KGE weight.

4.7. Sensitivity Analysis of the Model to Data Sparseness

Next, we conduct experimental analysis on the performance of the model in different sparse environments. The purpose is to analyze the model's ability to deal with data sparsity and alleviate cold-start problems. We conduct experimental comparisons under the M dataset. From the experimental results in Table 6, reduce the number of training sets to 30%, 60%, and 90% input model. It can be concluded that when the amount of data in the training set decreases, the performance of all models will decrease accordingly. In the amount of data decreases during training, the MKR model has better performance, and the AUC performance compared to the full training set decreases by 4.5%, 2.2%, and 0.7%, respectively, and ACC is 5.5%, 3.5%, and 0.1%. The AKNEI model corresponds to AUC reduction rates of 3.8%, 2.5%, and 0.4%, respectively, while the ACC is 4.3%, 3.1%, and 0.2%. The AKNEI still shows good performance, even when the data are sparse.

Table 6. Model performance under different scale M datasets.

Model	30%		60%		90%	
	AUC	ACC	AUC	ACC	AUC	ACC
LibFM	0.8165	0.7652	0.8501	0.7863	0.8864	0.8126
PER	0.6201	0.5842	0.6605	0.5889	0.7002	0.6118
Wide & Deep	0.8092	0.7542	0.8423	0.7818	0.8845	0.8105
DKN	0.5892	0.5231	0.6189	0.5767	0.6462	0.5997
RippleNet	0.8623	0.8012	0.8796	0.8155	0.9087	0.8305
MKR	0.8792	0.7952	0.8909	0.8123	0.9115	0.8313
KGCN	0.8766	0.8223	0.8902	0.8135	0.9089	0.8307
AKNEI	**0.8953**	**0.8131**	**0.9032**	**0.8234**	**0.9225**	**0.8464**

5. Conclusions and Future Work

The model proposed in this paper, AKNEI, is an end-to-end learning framework that combines a KG and neural networks. It obtains the implicit interaction of low-level features between KG entities and items through the ISL layer and uses the corrugated network and a cross feature fusion network layer for feature analysis. It has high-level display interaction. The KG layer plays an auxiliary recommendation role in the learning process and can be optimized for specific recommendation tasks. Through the evaluation of multiple public datasets, it can be seen that the recommendation performance of AKNEI in different scenarios is better than other baseline models. Our model can achieve good performance regardless of movie recommendations with large data volumes or music recommendations with small data volumes. This shows that AKNEI is suitable for recommendation tasks in a variety of different scenarios. The AKNEI model also shows good performance when addressing the data sparseness and cold-start problems that are common in recommender systems.

In future work, we will (1) use a nonuniform sampler to spread preferences to better tap the potential interests of users and (2) embed heterogeneous networks into the model framework to more fully extract KG semantic information.

Author Contributions: Conceptualization, J.Q.; Methodology, S.D.; Software, X.W.; Validation, S.D., X.W. and R. W.; Formal analysis, J.Q.; Investigation, X.W. and R.W.; Resources, J.Q.; Writing—original draft, S.D.; Writing—review & editing, R.W. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Science Fund for Outstanding Youth of Xinjiang Uygur Autonomous Region under Grant No. 2021D01E14, the National Science Foundation of China under Grant No. 61867006, The Major science and technology project of Xinjiang Uygur Autonomous Region under Grant No. 2020A03001 , the Innovation Project of Sichuan Regional under Grant No. 2020YFQ2018 and the Key Laboratory Open Project of Science & Technology Department of Xinjiang Uygur Autonomous Region named Research on Video Information Intelligent Processing Technology for Xinjiang Regional Security.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets generated and/or analyzed during the current study are available from the corresponding author on reasonable request.

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

References

1. Koren, Y.; Bell, R.; Volinsky, C. Matrix factorization techniques for recommender systems. *Computer* **2009**, *42*, 30–37. [CrossRef]
2. Jamali, M.; Ester, M. A matrix factorization technique with trust propagation for recommendation in social networks. In Proceedings of the fourth ACM conference on Recommender Systems, Barcelona, Spain, 26–30 September 2010; pp. 135–142.
3. Wang, H.; Zhang, F.; Hou, M.; Xie, X.; Guo, M.; Liu, Q. Shine: Signed heterogeneous information network embedding for sentiment link prediction. In Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, Marina Del Rey, CA, USA, 5–9 February 2018; pp. 592–600.

4. Sun, Y.; Yuan, N.J.; Xie, X.; McDonald, K.; Zhang, R. Collaborative intent prediction with real-time contextual data. *ACM Trans. Inf. Syst. (TOIS)* **2017**, *35*, 1–33. [CrossRef]
5. Huang, J.; Zhao, W.X.; Dou, H.; Wen, J.R.; Chang, E.Y. Improving sequential recommendation with knowledge-enhanced memory networks. In Proceedings of the The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, Ann Arbor, MI, USA, 8–12 July 2018; pp. 505–514.
6. Wang, H.; Zhang, F.; Wang, J.; Zhao, M.; Li, W.; Xie, X.; Guo, M. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management, Torino, Italy, 22–26 October 2018; pp. 417–426.
7. Wang, H.; Zhang, F.; Xie, X.; Guo, M. DKN: Deep knowledge-aware network for news recommendation. In Proceedings of the 2018 World Wide Web Conference, Lyon, France, 23–27 April 2018; pp. 1835–1844.
8. Yu, X.; Ren, X.; Sun, Y.; Gu, Q.; Sturt, B.; Khandelwal, U.; Norick, B.; Han, J. Personalized entity recommendation: A heterogeneous information network approach. In Proceedings of the 7th ACM International Conference on Web Search and Data Mining, New York, NY, USA, 24–28 February 2014; pp. 283–292.
9. Zhao, H.; Yao, Q.; Li, J.; Song, Y.; Lee, D.L. Meta-graph based recommendation fusion over heterogeneous information networks. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017; pp. 635–644.
10. Wang, Q.; Mao, Z.; Wang, B.; Guo, L. Knowledge graph embedding: A survey of approaches and applications. *IEEE Trans. Knowl. Data Eng.* **2017**, *29*, 2724–2743. [CrossRef]
11. Cheng, H.T.; Koc, L.; Harmsen, J.; Shaked, T.; Chandra, T.; Aradhye, H.; Anderson, G.; Corrado, G.; Chai, W.; Ispir, M.; et al. Wide & deep learning for recommender systems. In Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, Boston, MA, USA, 15 September 2016; pp. 7–10.
12. Misra, I.; Shrivastava, A.; Gupta, A.; Hebert, M. Cross-stitch networks for multi-task learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 July 2016; pp. 3994–4003.
13. Yosinski, J.; Clune, J.; Bengio, Y.; Lipson, H. How transferable are features in deep neural networks? *arXiv* **2014**, arXiv:1411.1792.
14. Guo, H.; Tang, R.; Ye, Y.; Li, Z.; He, X. DeepFM: A factorization-machine based neural network for CTR prediction. *arXiv* **2017**, arXiv:1703.04247.
15. Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; Yakhnenko, O. Translating embeddings for modeling multi-relational data. In Proceedings of the NIPS'13: Proceedings of the 26th International Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 5–10 December 2013, Volume 26.
16. Lin, Y.; Liu, Z.; Sun, M.; Liu, Y.; Zhu, X. Learning entity and relation embeddings for knowledge graph completion. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, Austin, TX, USA, 25–30 January 2015.
17. Liu, H.; Wu, Y.; Yang, Y. Analogical inference for multi-relational embeddings. In Proceedings of the International Conference on Machine Learning. PMLR, Sydney, Australia, 6–11 August 2017; pp. 2168–2178.
18. Nickel, M.; Rosasco, L.; Poggio, T. Holographic embeddings of knowledge graphs. In Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–14 February 2016; Volume 30.
19. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 5998–6008.
20. Wu, Y.; Schuster, M.; Chen, Z.; Le, Q.V.; Norouzi, M.; Macherey, W.; Krikun, M.; Cao, Y.; Gao, Q.; Macherey, K.; et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv* **2016**, arXiv:1609.08144.
21. Rendle, S.; Freudenthaler, C.; Schmidt-Thieme, L. Factorizing personalized markov chains for next-basket recommendation. In Proceedings of the 19th International Conference on World Wide Web, Raleigh, NC, USA, 26–30 April 2010; pp. 811–820.
22. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y. Graph attention networks. *arXiv* **2017**, arXiv:1710.10903.
23. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
24. Song, W.; Shi, C.; Xiao, Z.; Duan, Z.; Xu, Y.; Zhang, M.; Tang, J. Autoint: Automatic feature interaction learning via self-attentive neural networks. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, Beijing, China, 3–7 November 2019; pp. 1161–1170.
25. Kim, D.; Park, C.; Oh, J.; Lee, S.; Yu, H. Convolutional matrix factorization for document context-aware recommendation. In Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, 15–19 September 2016; pp. 233–240.
26. Zhang, W.; Du, T.; Wang, J. Deep learning over multi-field categorical data. In Proceedings of the European Conference on Information Retrieval, Padua, Italy, 20–23 March 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 45–57.
27. Qu, Y.; Cai, H.; Ren, K.; Zhang, W.; Yu, Y.; Wen, Y.; Wang, J. Product-based neural networks for user response prediction. In Proceedings of the 2016 IEEE 16th International Conference on Data Mining (ICDM), Barcelona, Spain, 12–15 December 2016; pp. 1149–1154.
28. Lian, J.; Zhou, X.; Zhang, F.; Chen, Z.; Xie, X.; Sun, G. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 1754–1763.

29. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed representations of words and phrases and their compositionality. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 5–10 December 2013; pp. 3111–3119.
30. He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; Chua, T.S. Neural collaborative filtering. In Proceedings of the 26th International Conference on World Wide Web, Perth, Australia, 3–7 April 2017; pp. 173–182.
31. Vig, J.; Sen, S.; Riedl, J. Tagsplanations: Explaining recommendations using tags. In Proceedings of the 14th International Conference on Intelligent User Interfaces, Sanibel Island, FL, USA, 8–11 February 2009; pp. 47–56.
32. Sharma, A.; Cosley, D. Do social explanations work? Studying and modeling the effects of social explanations in recommender systems. In Proceedings of the 22nd international conference on World Wide Web, Rio de Janeiro, Brazil, 13–17 May 2013; pp. 1133–1144.
33. Bauman, K.; Liu, B.; Tuzhilin, A. Aspect based recommendations: Recommending items with the most valuable aspects based on user reviews. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017; pp. 717–725.
34. Zhang, Y.; Lai, G.; Zhang, M.; Zhang, Y.; Liu, Y.; Ma, S. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval, Gold Coast, Australia, 6–11 July 2014; pp. 83–92.
35. Wang, H.; Zhang, F.; Zhao, M.; Li, W.; Xie, X.; Guo, M. Multi-task feature learning for knowledge graph enhanced recommendation. In Proceedings of the The World Wide Web Conference, San Francisco, CA, USA, 13–17 May 2019; pp. 2000–2010.
36. Ma, W.; Zhang, M.; Cao, Y.; Jin, W.; Wang, C.; Liu, Y.; Ma, S.; Ren, X. Jointly learning explainable rules for recommendation with knowledge graph. In Proceedings of the The World Wide Web Conference, San Francisco, CA, USA, 13–17 May 2019; pp. 1210–1221.
37. Rendle, S. Factorization machines with libfm. *ACM Trans. Intell. Syst. Technol. (TIST)* **2012**, *3*, 1–22. [CrossRef]
38. Wang, H.; Zhao, M.; Xie, X.; Li, W.; Guo, M. Knowledge graph convolutional networks for recommender systems. In *The World Wide Web Conference*; ACM: New York, NY, USA, 2019; pp. 3307–3313.
39. Wang, Y.Q.; Dong, L.Y.; Li, Y.L.; Zhang, H. Multitask feature learning approach for knowledge graph enhanced recommendations with RippleNet. *PLoS ONE* **2021**, *16*, e0251162. [CrossRef] [PubMed]
40. Huang, W.; Wu, J.; Song, W.; Wang, Z. Cross attention fusion for knowledge graph optimized recommendation. *Appl. Intell.* **2022**, *52*, 10297–10306. [CrossRef]