Article

A Knowledge Graph-Enhanced Attention Aggregation Network for Making Recommendations

Dehai Zhang , Xiaobo Yang, Linan Liu and Qing Liu *

College of Software, Yunnan University, Kunming 650504, China; dhzhang@ynu.edu.cn (D.Z.);
xb_yang@mail.ynu.edu.cn (X.Y.); liulinan@mail.ynu.edu.cn (L.L.)

* Correspondence: liuqing@ynu.edu.cn

Abstract: In recent years, many researchers have devoted time to designing algorithms used to introduce external information from knowledge graphs, to solve the problems of data sparseness and the cold start, and thus improve the performance of recommendation systems. Inspired by these studies, we proposed KANR, a knowledge graph-enhanced attention aggregation network for making recommendations. This is an end-to-end deep learning model using knowledge graph embedding to enhance the attention aggregation network for making recommendations. It consists of three main parts. The first is the attention aggregation network, which collect the user's interaction history and captures the user's preference for each item. The second is the knowledge graph-embedded model, which aims to integrate the knowledge. The semantic information of the nodes and edges in the graph is mapped to the low-dimensional vector space. The final part is the information interaction unit, which is used for fusing the features of two vectors. Experiments showed that our model achieved a stable improvement compared to the baseline model in making recommendations for movies, books, and music.

Keywords: knowledge graph; personalized recommendation; attention aggregation

Citation: Zhang, D.; Yang, X.; Liu, L.; Liu, Q. A Knowledge Graph-Enhanced Attention Aggregation Network for Making Recommendations. *Appl. Sci.* **2021**, *11*, 10432. <https://doi.org/10.3390/app112110432>

Academic Editors:
Ángel González-Prieto and
Fernando Ortega

Received: 21 July 2021
Accepted: 28 October 2021
Published: 5 November 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In many online services such as e-commerce, Internet advertising, and social media, people access online content (usually in the form of purchases or clicks), thus generating a large number of interactive records. To reduce the impact of information overload, researchers have proposed recommendation systems to satisfy the personalized needs of users. Traditional recommendation methods, such as collaborative filtering (CF) and matrix factorization (MF) [1], predict whether users are interested in an item based on their historical interactions. However, these methods usually suffer from the problems of data sparsity and the cold start. Researchers have introduced external information to solve these problems; for example, social networks [2,3], item attributes [4], knowledge graphs [5], and other heterogeneous networks. Knowledge graphs are widely used in recommendation tasks because of their high-order connectivity and sufficient prior knowledge. In the recommendation task based on the knowledge graph, users and items correspond to nodes in the graph, and the relationships between items correspond to edges in the graph. At present, many research institutions open-source their academic knowledge graphs, such as DBpedia [6] and Google Knowledge Graph [7].

Inspired by these methods, we consider that the semantic relationship between users and items can improve the effect of recommendations. For an item, different users have different interactive behaviors, which represent the user's preference. For example, a user's interaction behavior in relation to an iPad Pro is "purchase", whereas another's interaction behavior is "browse". Obviously, compared with browsing, the interaction behavior of purchasing indicates the user prefers the iPad Pro. Therefore, we take users and items as entities, and the interaction behavior as the relationship between users and items. This forms a user-item interaction knowledge graph, as shown in Figure 1.

Figure 1. Illustration of the user-item interaction knowledge graph. The users' historical interaction records can be used to construct a user-item interaction knowledge graph. The interaction behavior can be regarded as the semantic relationship between nodes.

Most researchers divide the recommendation system (RS) [8] and knowledge graph embedding (KGE) [9] into two independent tasks and train them in two different vector spaces. This method is convenient for training different vectors. The error caused by spaces will reduce the overall performance of the model. We concede that recommendation systems and knowledge graph embedding are not completely independent tasks [10]. Inspired by multitask learning [11], we propose KANR, an end-to-end deep learning model using knowledge graph embedding to enhance the attention aggregation network for making recommendations. The main contributions of this work are as follows:

- We designed an information interaction unit to combine a recommendation system and knowledge graph embedding tasks. KANR can learn the semantic information in the knowledge graph in a unified vector space.
- We proposed an attention aggregation network which is used to collect users' interaction history and mine users' preferences to improve personalized recommendations.
- We conducted a series of experiments on three open-source datasets to verify the effectiveness of KANR. The results showed that KANR more effectively learned user preferences and performed well in click-through rate prediction and Top-K recommendation tasks.

2. Related Work

2.1. Recommendation Systems

At present, recommendation system based on knowledge graphs can be divided into three types: embedding-based methods, path-based methods, and graph-based methods.

The embedding-based method uses KGE to map the items and relationships in the knowledge graph to a low-dimensional vector space, and to design a model for learning the features of users and items. The Deep Knowledge-aware network (DKE) [12] first pre-trains the entities in the knowledge graph and then uses a VGG network [13] to extract the features of the joint vector of the entity and the words in news headlines. This method uses a simple framework to integrate the entities of the knowledge graph with news recommendations. However, this method only uses the words in the title to make recommendations, so the corresponding context information is missing. Collaborative knowledge base embedding for recommender (CKE) [14] uses three embedding methods to fit the knowledge graph, text, and visual information in the knowledge base. Then, it combines the three vectors and the implicit feedback of users in the recommendation system.

Multiple paths connect the nodes in the knowledge graph. These paths can provide more additional information and interpretability for the recommendation system. The path-based method uses the path information in the knowledge graph to enhance the recommendation system and provide explanations for the recommendation results. Explainable Reasoning over Knowledge Graphs for Recommendation (KPRN) [15] uses a long short-term memory (LSTM) network [16] to capture the dependencies and inference paths of items to infer user preferences and generate reasonable explanations. Although the model based on path traversal provides interpretability and a good performance improvement, it consumes a large amount of computing resources when learning the optimal path. RuleGuider [17] proposed a rule guider to learn the probability distribution of the reasoning path. RuleGuider uses a symbol-based method to mine high-quality rules, and introduces an agent in reinforcement learning [18] to learn. The walking path uses high-quality rules to provide reward supervision for the agent.

Graph-based methods focus on the association between nodes and neighbors in the knowledge graph. A common approach is to treat a specific entity node as a center for aggregation to capture the node's characteristics and neighbors. Knowledge Graph Convolutional Networks for Recommender Systems (KGCN) [19] uses a graph convolutional network to aggregate neighbor nodes to obtain the k-hop structure information of the central node. Simultaneously, it weights neighbors according to the connection relationship and specific user scores to represent the semantic information of the knowledge graph. Neighborhood Aggregation Collaborative Filtering Based on Knowledge Graph (NACF) [20] uses the attention in the graph convolutional network (GCN) [21], calculates the weight for historical interactive items, and then uses the graph convolutional network to aggregate the items to capture the user's preferences.

2.2. Knowledge Graph Embedding

The function of knowledge graph embedding is to embed the entity and relationship in the knowledge graph into a continuous low-dimensional vector space to facilitate the sharing of the features in the knowledge graph.

At present, the mainly used knowledge graph embedding models are divided into two types. The first is the translation model; for example TransR [22] and TransD [23], can represent the structural information of the knowledge graph. TransR maps entities into R relational space through matrix M_r , and learns embedding through $h_r + r = t_r$ in different spaces. TransD sets up two matrices to map the head entity and tail entity into the relational space, respectively, so that the head entity and tail entity can support different embedding under the same relationship. The second is the semantic matching model [24,25]; for example, the Deep Structured Semantic Model (DSSM) [26] and DistMult [27], can represent the semantic information in the knowledge graph. DSSM maps h and t into the semantic space of common dimensions, and trains the implicit semantic model by maximizing the cosine similarity between h and t semantic vectors. DistMult is a simplified version of a latent factor model (LFM) that limits M_r to a diagonal matrix and effectively learns the semantic relationship between entities.

3. Methods

3.1. Formulation

Inspired by multi-task learning, KANR uses the information interaction unit to combine the two tasks of RS and KGE. In the recommendation task, we define a set of N users $U = \{u_1, u_2, \dots, u_n\}$ and a set of M items $I = \{i_1, i_2, \dots, i_m\}$. The set of user's historical interactive items are defined as $V_i = (v_1, v_2, \dots, v_K)$, where $V_i \in I$ represents the historical interactive items of u_i . In the pre-training of the knowledge graph, we define a proprietary knowledge graph for each dataset. The knowledge graph is expressed as (h, r, t) , where h and t represent the head entity and the tail entity, respectively, and r represents the relationship between entities. In the research of recommendation systems based on the knowledge graph, the entities are regarded as items and the relationship between the enti-

ties is equivalent to the attributes between items. Through the path link in the knowledge graph, the recommendation system can obtain more side information.

3.2. Our Model

The framework process of KANR is shown in Figure 2a. First, a user vector u and the set of the user's historical interactive items V are input. u and V are aggregated into the user-item interaction vector e through the attention aggregation network. The attention model can calculate the value of each item weight, so e contains the user's preference for historical interactive items. Then, the model calculates the scores of items i and e for the recommendation. The right half is the knowledge graph embedding. The middle part is the information interaction model, which fuses the features between e and t and the features between I and h .

Figure 2. (a) The framework of KANR. The left-hand side shows the attention aggregation network. The user and interaction item vector is aggregated into e . The right-hand side shows the knowledge graph embedding model. After feature fusion through the Information Interaction model, e and i are regarded as t and h to predict the semantic relationship. At the same time, e and i are connected to calculate the recommendation confidence. (b) The attention model in KANR. After feature fusion through the Information Interaction model, the user and an interacted item vector is connected to calculate the attention.

3.2.1. Information Interaction Unit

The information interaction unit is a model that fuses features for two vectors. Inspired by the Deep and Cross Network (DCN) [28], we use feature cross and feature compression to complete the information interaction. Its framework is shown in Figure 3.

Figure 3. Information Interaction Unit. There are two vectors, ke and re . Feature cross is used to construct a d -dimensional matrix, and W_{ke} and W_{re} map a d -dimensional matrix to a d -dimensional vector.

Given the two input vectors ke and re , we first multiply ke and re to construct a feature cross matrix C :

$$C = ke \cdot re^T = \begin{bmatrix} ke_1 re_1 & \cdots & ke_1 re_d \\ \vdots & \ddots & \vdots \\ ke_d re_1 & \cdots & ke_d re_d \end{bmatrix}, \quad (1)$$

where $C \in \mathbb{R}^{d \times d}$ is the cross matrix. Because each possible combination $ke_i re_j \forall (i, j) \in \{1, 2, \dots, d\}^2$ between all elements, C has completed the feature cross between vectors. Then, we cross the features as the input vector of the recommendation module and the knowledge graph embedding module:

$$ke' = ke + CW_{ke} + b_{ke}, \quad (2)$$

$$re' = re + CW_{re} + b_{re}, \quad (3)$$

where $W_{ke}, W_{re}, b_{ke}, b_{re}$ represent trainable weights and biases. W_{re} and W_{ke} map the interaction matrix C into a d -dimensional vector space which generates new feature mixed vectors ke' and re' . After the vectors are mixed, they share each other's features. The information interaction model is described by $\mathcal{M}_C$.

$$ke', re' = \mathcal{M}_C(ke, re). \quad (4)$$

3.2.2. Attention Aggregation Network

The attention aggregation network calculates the aggregation vector e of the user and historical interactive items, which we call the user-item interaction vector.

We use the user vector u and the historical interactive item vector v to calculate the user-item aggregation vector. In order to mine user preferences, we propose a novel attention model to assign item weights (as shown in Figure 2b). This combines the user vector u_i with the history interactive item vector v_{ij} , which share features with each other, and then connects these two vectors to calculate the scores. The score is the weight of the item, and also indicates the user's preference. The calculation process is described in the following formulas.

We first use the information interaction model to share the features between the initial user vector and the item vector. u_i', v_{ij}' are the mixed feature vectors of the user and the item, which are expressed in the use of the attention model for each history. The interactive item is assigned the weight W_{ij} , which represents the weight of the user's historical interactive item. We then calculate the weighted average of the historical interactive item vector and the initial user vector to obtain the user-item aggregation vector e .

$$u_i', v_{ij}' = \mathcal{M}_C(u_i, v_{ij}), \quad (5)$$

$$W_{ij} = \text{Softmax}(\text{ReLU}(W_{att}(u_i' || v_{ij}') + b_{att})), \quad (6)$$

$$e = u_i + \frac{1}{k} \sum_{j=1}^k W_{ij} \dots v_{ij}. \quad (7)$$

3.2.3. Prediction

First, we use the information interaction unit to process the head entity h and the user-item aggregation vector e_i to generate h' and e' . Furthermore, the tail entity t and new item i are also processed. Next, the head entity h' is combined with the tail entity t' to calculate the relation vector $\hat{r}_{ht}$.

$$e', h' = \mathcal{M}_C(e, h), \quad (8)$$

$$i', t' = \mathcal{M}_C(i, t), \quad (9)$$

$$\hat{r}_{ht} = \delta(W_k((h' || t') + b)), \quad (10)$$

where $W_k \in \mathbb{R}^{2d}$ are trainable weights and $||$ represents a connection. δ is a nonlinear transformation. Then, we calculate the $Score(h, r, t)$ between them using the distance function. Finally, e' and i' are multiplied to calculate $\hat{y}_{ui}$:

$$Score(h, r, t) = ||\hat{r}_{ht} - r_{ht}||_2^2, \quad (11)$$

$$\hat{y}_{ui} = \delta(e'_i \odot i'). \quad (12)$$

3.3. Learning

We initialize all users and items in the same vector space, obtain their low-dimensional vector representation, and then use the attention aggregation network to aggregate users and interactive historical items to obtain the user-item aggregation vector. We then use the information interaction model and the nodes of the knowledge graph to share features, and finally calculate the recommendation score and KGE score. After N epochs, KANR gradually learns the optimization results.

In order to better learn the parameters of KANR, we designed the following complete loss function:

$$\begin{aligned} L &= L_{RS} + L_{KGE} + L_2 \\ &= \sum_{u \in U, v \in V} J(\hat{y}_{uv}, y_{uv}) \\ &\quad - \lambda_1 \left(\sum_{(h,r,t) \in G} score(h, r, t) - \sum_{(h,r',t) \notin G} score(h, r', t) \right) + \lambda_2 ||w||_2^2 \end{aligned} \quad (13)$$

The complete loss function of KANR is formed by adding the three parts of L_{RS} , L_{KGE} , and L_2 , which respectively express the loss function of the recommendation model, the knowledge graph embedding model, and L_2 regularization. In L_{RS} , we use the cross-entropy function J as the loss function of the recommendation model. In L_{KGE} , we calculate the score difference between the positive sampling and the negative sampling of the triple as the confidence of the embedding result, so as to improve the effect of the model. Finally, we add L_2 regularization to the complete loss function to prevent the model from overfitting, where λ_1 and λ_2 are hyperparameters.

4. Experiment and Results

4.1. Data and Experimental Environment

In this study, we used three datasets: MovieLens-1M, Last.FM, and Book-Crossings.

MovieLens-1M comprises about 1 million clear scoring datapoints (scoring from 1 to 5) on the MovieLens website.

Last.FM collects the rating data of 2000 users in the online music system (from 1 to 352,698). The corresponding KG contains 9366 entities, 15,518 edges, and 60 relationship types.

Book-Crossings is a book scoring dataset. It contains 1.1 million ratings for 270,000 books from 90,000 users. The score ranges from 1 to 10, including explicit and implicit scores. The Book-Crossings dataset is one of the least dense datasets, and is also the least dense dataset with a clear score.

The user-item interaction graph corresponds to the dataset samples from the click matrix. In data preprocessing, we treat users and historical interaction items as entities, and the interaction behavior as relations. Because MovieLens-1M and Last.FM comprise scoring feedback, we use manually set rules to convert scoring feedback into click feedback. In the movie dataset, we selected records with 4 points and above (full score is 5) as positive samples. In the music dataset, because the music scores are too sparse (from 1 to 352,698), they cannot be used as a criterion for evaluating user interests. Therefore, we set all the music clicked by the user as a positive sample. Similarly, all interactive items (including 0 to 10 points) in Book-Crossing were regarded as positive samples. In order to avoid too large a gap between the number of positive samples and the number of negative samples, we adopted a negative sample method during training. This method randomly selected

negative samples from items in the dataset without user interaction until the numbers of positive samples and negative samples were equal. In addition, we treated different ratings as different types of interaction. In Movie-1M and Book-Crossing, we set up five and 10 interaction types in the user–item interaction graph.

Table 1 shows the basic statistics of the three datasets and the hyperparameters of KANR. λ_1 and λ_2 were both set to 0.01 and 0.001. (D represents the embedding vector dimension of the entity, and N represents the number of historical interactive items of the user). KANR was implemented under the Windows 10 operating system, using Python 3.7, Tensorflow-gpu 1.12.0, cudnn 7.1.4, and NumPy 1.15.4. The experimental hardware environment was AMD R5 2600, GTX2070TI, and 16G memory. In this experiment, the training set, evaluation set, and test set were set to a ratio of 6:2:2, and the hyperparameters of the model were dynamically adjusted according to the AUC index.

Table 1. Basic data statistics of the dataset and hyperparameter settings.

Dataset	#Users	#Items	#Triples	#Interaction Types	D	N	Batch Size
Movie-1M	6036	2347	20,195	5	64	4	512
Last.FM	1872	3846	15,518	12	16	5	128
Book-Crossings	17,860	14,910	19,793	10	16	4	128

4.2. Baseline

We selected seven baseline models to compare the performance of KANR on three datasets, including three recommendation algorithms combining knowledge graphs and two classic recommendation algorithms. The baseline model was as follows:

Wide&Deep [29] proposed a recommendation algorithm that combines a deep model and a shallow wide model for fusion training. The embedding vector dimensions of users and items in Wide&Deep are unified to 64, and a double-layer deep channel with dimensions of 100 and 50 and a wide channel are set at the same time.

CKE [14] proposed a unified recommendation framework to embed multimodal data such as knowledge graphs, text information, and picture information into the recommendation task. This paper sets the embedding vector dimensions of CKE users and items in three datasets. The dimensions are 64, 128, and 32, and the entity’s vector embedding dimension is uniformly set to 32.

LibFM [30] is a feature-based factorization model widely used for CTR prediction. In this paper, TransR is used to train the initial user and item vectors.

RippleNet [5] is a hybrid method that uses the knowledge graph structure to assist with making recommendations. It completes personalized recommendation tasks by exploring the user’s potential interest characteristics on the knowledge graph. This paper uses the TransE [14] algorithm to learn the embedding vectors of users and items.

KGCN [19] is a recommendation algorithm that uses knowledge graphs to convolve item features. The graph convolution operation greatly increases the utilization efficiency of the knowledge graph’s network structure information. This paper sets the entity embedding vector dimensions of KGCN in the three datasets to be 32, 64 and 16, and the number of neighbor nodes is 8.

MKR [10] proposed a multi-task feature learning method combining knowledge embedding and recommendation tasks. By combining the knowledge graph embedding algorithm and the recommendation system module, the potential information of the recommended scene and the knowledge graph can be exchanged.

GMCF [31] proposed a collaborative filtering model based on neural network graph matching. By modeling and aggregating the attribute interaction in the graph matching structure, two types of attribute interaction are effectively captured.

4.3. Results

In order to comprehensively test the performance of KANR in the recommended scenario, in this study we conducted experiments on the two tasks of Top-K recommendation and CTR prediction, and compared the results of KANR with the above-mentioned baseline model. All experiments were performed four times, and the average value of the index was calculated.

4.3.1. Metrics

In CTR prediction, we used Area Under Curve (AUC) and Accuracy (ACC) to evaluate the performance of all models. AUC can still make a reasonable evaluation of the classifier in the case of unbalanced samples. ACC describes how many of the predicted positive examples were true. In the Top-K recommendation, we used Recall@K and Precision@K as metrics. Recall@K refers to the ratio of the number of Top-K results to the number of all relevant results. Precision@K quantifies how many of the Top-K results were relevant.

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \quad (14)$$

$$Recall@K = \frac{TP@K}{TP@K + FN@K} \quad (15)$$

$$Precision@K = \frac{TP@K}{TP@K + FP@k} \quad (16)$$

FP is the number of negative examples predicted incorrectly. TN is number of negative examples predicted correctly. FN is the number of positive examples predicted incorrectly.

4.3.2. The Performance in CTR Prediction

In CTR prediction, we used AUC (Area Under Curve) and ACC (Accuracy) to evaluate the performance of all models. In addition, we also evaluated the enhancement effect of the knowledge graph embedding model and the attention model on KANR, as shown in Table 2.

Table 2. The performance of each model in CTR prediction. The best results are in bold.

Method	MovieLens-1M		Last.FM		Book-Crossing	
	AUC	ACC	AUC	ACC	AUC	ACC
Wide&Deep	0.898 (−3.0%)	0.820 (−3.6%)	0.756 (−7.4%)	0.688 (−8.5%)	0.712 (−4.3%)	0.624 (−11.1%)
CKE	0.801 (−13.7%)	0.742 (−12.8%)	0.744 (−8.9%)	0.673 (−10.5%)	0.671 (−9.8%)	0.673 (−4.2%)
LibFM	0.892 (−3.9%)	0.812 (−4.5%)	0.777 (−4.8%)	0.709 (−5.7%)	0.685 (−7.9%)	0.640 (−8.6%)
Ripple	0.920 (−0.9%)	0.842 (−1.1%)	0.780 (−4.5%)	0.702 (−6.6%)	0.729 (−2.0%)	0.662 (−5.6%)
MKR	0.924 (−0.5%)	0.848 (−0.3%)	0.796 (−2.5%)	0.752 (0%)	0.738 (−0.8%)	0.688 (−1.9%)
KGCN	0.917 (−1.3%)	0.843 (−0.9%)	0.796 (−2.5%)	0.728 (−3.2%)	0.731 (−1.7%)	0.678 (−3.4%)
GMCF	0.918 (−1.1%)	0.845 (−0.7%)	0.785 (−3.9%)	0.71 (−2.3%)	0.789 (+6.0%)	0.712 (+1.4%)
KANR	0.929	0.851	0.817	0.752	0.744	0.702
KANR-K	0.903 (−2.7%)	0.826 (−2.9%)	0.782 (−4.3%)	0.742 (−3.5%)	0.718 (−3.5%)	0.668 (−4.8%)
KANR-A	0.921 (−0.8%)	0.839 (−1.4%)	0.808 (−1.1%)	0.746 (−0.8%)	0.726 (−2.4%)	0.679 (−3.2%)

KANR-KGE means that the model only trained the recommendation model and did not interact with the knowledge graph embedded model. KANR-ATT indicates that the model only used the average aggregation algorithm in the aggregation process of the user vector and the historical interaction item vector, and did not add the item weight calculated by the attention model.

In CTR prediction, KANR slightly leads the baseline model in terms of AUC and ACC. In the movie dataset, KANR has certain performance advantages, whereas the AUC index in the music and book datasets has been significantly improved. However, it also lags slightly behind in the accuracy index of the book dataset. In addition, we can see

that by excluding the knowledge graph embedding module and the attention module, the performance of the three datasets shows different degrees of decline, which proves that the semantic information in the knowledge graph and the attention model provides a certain improvement in performance for the recommendation task. As a result of the enhancement of KANR, the information interaction unit of KANR can effectively share vector features.

4.3.3. The Performance in Top-K Recommendation

In the Top-K recommendation task, we recommend the K items with the highest matching degree for each user in the dataset after the model training is completed. Recall@K and Precision@K ($K = 1, 2, 5, 10, 50, 100$) are used as evaluation indicators to evaluate the performance of each model.

As shown in Figure 4, KANR achieved the best performance in the Top-K recommendations of the three datasets. In Recall@10, the performance of KANR on the movie, music, and book datasets was 6.04%, 11.38%, and 14.03% higher than the best baseline. In Precision@10, it also achieved a performance gain of 9.09%, 7.6%, and 16.91%, which shows that KANR can effectively use the semantic information in the knowledge graph to enhance the recommendation model.

Figure 4. Performance display of each model in the Top-K experiment.

In general, the performance of the model based on the knowledge graph is superior to that of the traditional model. In addition, the performance of KANR, which uses the multi-task learning framework and information interaction model, is superior to that of other types of model, especially in the book dataset. The reason for this is that, in the case of the information interaction model, the knowledge triples in the dataset. The vector features can still be shared, and the recommendation system obtains effective semantic information in the knowledge graph and is not affected by knowledge sparseness.

4.3.4. The Performance in the Cold Start Environment

This section verifies whether KANR can effectively alleviate the cold start problem. We simulated the cold start environment by adjusting the proportion of the training set. When $r = 20\%$, the AUC of the five baselines decreased by 7.12%, 6.23%, 4.91%, 2.01%, and 2.42%, respectively. The performance of KANR only dropped by 1.5%. The results of the model under the AUC evaluation index are shown in Figure 5. This shows that KANR can achieve better results than the baseline in cold start scenarios. Similarly, in this

experiment, the recommendation strategy combined with the knowledge graph performed better than the traditional recommendation methods such as Wide&Deep. This proves that the side information of the knowledge graph can alleviate the cold start problem of the collaborative filtering model.

Figure 5. Performance analysis of the model in a cold start scenario.

4.3.5. The Performance under Different Embedding Dimensions

This experiment analyzed the impact of different embedding dimensions of entities on the performance of KANR. The performance of the three datasets is shown in Table 3. With other parameters unchanged, the AUC of KANR in the movie dataset gradually increased with the increase in the d dimension, and reached the optimal performance in the case of 64 dimensions. In the music and book datasets, the model achieved the best performance in 16 dimensions. However, as the dimension increased, the performance gradually decreased. This shows that the entity embedding dimension adapted to each dataset can obtain the most effective data characteristics.

Table 3. The impact of different embedding dimensions on AUC indicators. The best results are in bold.

d	8	16	32	64	128
Movielens-1M	0.887	0.902	0.915	0.929	0.923
Last.FM	0.808	0.817	0.812	0.796	0.773
Book-Crossing	0.731	0.744	0.729	0.722	0.718

5. Discussion

The key factor in the recommendation task is the embedding of users and items. The CF-based method has the disadvantage of insufficient features due to sparse data. We found that the semantic information in the knowledge graph can effectively improve the effect of recommendations. Therefore, we propose KANR, an end-to-end recommendation method that uses semantic information in the user–item interaction knowledge graph to enhance the attention. It contains three parts, namely, a recommendation model, a knowledge graph embedding model, and an information interaction unit. The recommendation model uses the attention aggregation network to calculate the weights of historical interactive items and aggregate the product vectors to capture the user’s interest preferences. The knowledge graph embedding model takes the item to be recommended as the head entity and the user as the tail entity, and uses the Semantic Matching Energy Model (SME) to obtain their semantic vector. Both the recommendation model and the knowledge graph embedding model share vector features through the information interaction unit. The item vector to be recommended and the head entity vector share features, and the user–item aggregation vector and the tail entity vector share features. It is worth mentioning that KANR changes the traditional attention mechanism and uses the information interaction model to share user vectors and historical interaction item vectors. The advantage of this is

that it can introduce knowledge learned during information interaction into the attention mechanism. We conducted a series of experiments to prove that the semantic information in the knowledge graph can effectively enhance the recommendation model through the information interaction model.

6. Conclusions

The interaction between a user and an item represents the user's preference. We found that the semantic information of interaction can be used to improve the effect of the recommendation system. We proposed KANR and proved its effectiveness through experiments. Based on the research of this paper, we believe that the semantic relationship of the knowledge graph has great value in recommendation systems.

In the future, we will aim to design a more effective information interaction unit to combine with the recommendation model and the knowledge graph embedding model, and test the recommendation performance of different knowledge graph embedding models. In addition, we will also explore combining the knowledge graph with the recommendation problem in the sequence recommendation problem to realize a recommendation method using knowledge graph reasoning.

Author Contributions: Conceptualization, Q.L. and D.Z.; methodology, X.Y.; software, L.L. and X.Y.; validation, X.Y.; formal analysis, L.L.; investigation, X.Y.; writing—original draft preparation, X.Y.; writing—review and editing, L.L. and X.Y.; visualization, X.Y.; supervision, Q.L. and D.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by (i) Natural Science Foundation China (NSFC) under Grant No. 61402397, 61263043, 61562093 and 61663046; (ii) Open Foundation of Key Laboratory in Software Engineering of Yunnan Province: 2020SE304.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. MovieLens can be obtained from <https://grouplens.org/datasets/movielens/> (accessed on 27 October 2021). Last.FM can be obtained from <https://www.last.fm/> (accessed on 27 October 2021). Book-Crossing can be obtained from <http://www2.informatik.uni-freiburg.de/~cziegler/BX/> (accessed on 27 October 2021).

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

References

1. Rendle, S.; Krichene, W.; Zhang, L.; Anderson, J. Neural collaborative filtering vs. matrix factorization revisited. In Proceedings of the Fourteenth ACM Conference on Recommender Systems, Virtual Event, Brazil, 22–26 September 2020; pp. 240–248.
2. Song, W.; Xiao, Z.; Wang, Y.; Charlin, L.; Zhang, M.; Tang, J. Session-based social recommendation via dynamic graph attention networks. In Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, Melbourne VIC, Australia, 11–15 February 2019; pp. 555–563.
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. Dong, B.; Zhu, Y.; Li, L.; Wu, X. Hybrid collaborative recommendation of co-embedded item attributes and graph features. *Neurocomputing* **2021**, *442*, 307–316. [CrossRef]
5. 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.
6. Bizer, C.; Lehmann, J.; Kobilarov, G.; Auer, S.; Becker, C.; Cyganiak, R.; Hellmann, S. DBpedia—A crystallization point for the Web of Data. *J. Web Semant.* **2009**, *7*, 154–165. [CrossRef]
7. Pelikánová, Z. Google Knowledge Graph. 2014. Available online: https://dspace.muni.cz/handle/ics_muni_cz/1024 (accessed on 27 October 2021). (In Czech).
8. Das, D.; Sahoo, L.; Datta, S. A survey on recommendation system. *Int. J. Comput. Appl.* **2017**, *160*. Available online: <https://www.ijcaonline.org/archives/volume160/number7/das-2017-ijca-913081.pdf> (accessed on 27 October 2021). [CrossRef]

9. 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. [\[CrossRef\]](#)
10. Wang, H.; Zhang, F.; Zhao, M.; Li, W.; Xie, X.; Guo, M. Multi-task feature learning for knowledge graph enhanced recommendation. In Proceedings of the World Wide Web Conference, San Francisco, CA, USA, 13 May 2019; pp. 2000–2010.
11. Zhang, Y.; Yang, Q. A survey on multi-task learning. *arXiv* **2017**, arXiv:1707.08114.
12. 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 on World Wide Web, International World Wide Web Conferences Steering Committee, Lyon, France, 10 April 2018; pp. 1835–1844.
13. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2014**, arXiv:1409.1556.
14. Zhang, F.; Yuan, N.J.; Lian, D.; Xie, X.; Ma, W.Y. Collaborative knowledge base embedding for recommender systems. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13 August 2016; pp. 353–362.
15. Wang, X.; Wang, D.; Xu, C.; He, X.; Cao, Y.; Chua, T.S. Explainable reasoning over knowledge graphs for recommendation. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 5329–5336. [\[CrossRef\]](#)
16. Greff, K.; Srivastava, R.K.; Koutník, J.; Steunebrink, B.R.; Schmidhuber, J. LSTM: A search space odyssey. *IEEE Trans. Neural Netw. Learn. Syst.* **2016**, *28*, 2222–2232. [\[CrossRef\]](#) [\[PubMed\]](#)
17. Lei, D.; Jiang, G.; Gu, X.; Sun, K.; Mao, Y.; Ren, X. Learning Collaborative Agents with Rule Guidance for Knowledge Graph Reasoning. *arXiv* **2020**, arXiv:2005.00571.
18. Arulkumaran, K.; Deisenroth, M.P.; Brundage, M.; Bharath, A.A. Deep reinforcement learning: A brief survey. *IEEE Signal Process. Mag.* **2017**, *34*, 26–38. [\[CrossRef\]](#)
19. Wang, H.; Zhao, M.; Xie, X.; Li, W.; Guo, M. Knowledge graph convolutional networks for recommender systems. In Proceedings of the World Wide Web Conference, San Francisco, CA, USA, 13 May 2019; pp. 3307–3313.
20. Zhang, D.; Liu, L.; Wei, Q.; Yang, Y.; Yang, P.; Liu, Q. Neighborhood Aggregation Collaborative Filtering Based on Knowledge Graph. *Appl. Sci.* **2020**, *10*, 3818. [\[CrossRef\]](#)
21. Kipf, T.N.; Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv* **2016**, arXiv:1609.02907.
22. Lin, Y.; Liu, Z.; Sun, M.; Liu, Y.; Zhu, X. Learning entity and relation embeddings for knowledge graph completion. In Proceedings of the AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 19 February 2015; Volume 29.
23. Ji, G.; He, S.; Xu, L.; Liu, K.; Zhao, J. Knowledge graph embedding via dynamic mapping matrix. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, Beijing, China, 26–31 July 2015; Volume 1, pp. 687–696. Available online: <https://aclanthology.org/P15-1067.pdf> (accessed on 27 October 2021).
24. Liu, H.; Wu, Y.; Yang, Y. Analogical Inference for MultiRelational Embeddings. In Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; pp. 2168–2178. Available online: <http://proceedings.mlr.press/v70/liu17d/liu17d.pdf> (accessed on 27 October 2021).
25. Nickel, M.; Rosasco, L.; Poggio, T. Holographic Embeddings of Knowledge Graphs. In Proceedings of the 30th AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 7 December 2016; pp. 1955–1961.
26. Huang, P.S.; He, X.; Gao, J.; Deng, L.; Acero, A.; Heck, L. Learning deep structured semantic models for web search using clickthrough data. In Proceedings of the 22nd ACM international conference on Information & Knowledge Management, San Francisco, CA, USA, 27 October 2013; pp. 2333–2338.
27. Yang, B.; Yih, W.-T.; He, X.; Gao, J.; Deng, L. Embedding entities and relations for learning and inference in knowledge bases. In Proceedings of the International Conference on Learning Representations, San Diego, CA, USA, 7–9 May 2015.
28. Wang, R.; Fu, B.; Fu, G.; Wang, M. Deep & Cross Network for Ad Click Predictions. In Proceedings of the ADKDD'17, Halifax, NS, Canada, 14 August 2017.
29. Cheng, H.T.; Koc, L.; Harmsen, J.; Shaked, T.; Chandra, T.; Aradhye, H.; Anderson, G.; Corrado, G.; Chai, W.; Ispir, M. Wide & deep learning for recommender systems. In Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, Boston, MA, USA, 15 September 2016.
30. Steffen, R. Factorization machines with libfm. *ACM Trans. Intell. Syst. Technol.* **2012**, *3*, 1–22. [\[CrossRef\]](#)
31. Su, Y.; Zhang, R.; Erfani, S.; Gan, J. Neural Graph Matching based Collaborative Filtering. *arXiv* **2021**, arXiv:2105.04067.