



## Article

# Enhancing Knowledge of Propagation-Perception-Based Attention Recommender Systems

Hanzhong Zhang <sup>1,2</sup>, Yinglong Wang <sup>3,\*</sup>, Chao Chen <sup>2</sup>, Ruixia Liu <sup>2</sup>, Shuwang Zhou <sup>1,2</sup> and Tianlei Gao <sup>1,2</sup>

<sup>1</sup> College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China; zhanghz178@163.com (H.Z.); zhoushw@sdsas.org (S.Z.); gaotl@sdsas.org (T.G.)

<sup>2</sup> Shandong Artificial Intelligence Institute, Qilu University of Technology (Shandong Academy of Sciences), Jinan 266590, China; chench@sdsas.org (C.C.); liurx@sdsas.org (R.L.)

<sup>3</sup> Shandong Computer Science Center (National Supercomputer Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences), Jinan 266590, China

\* Correspondence: wangylscsc@126.com

**Abstract:** Researchers have introduced side information such as social networks or knowledge graphs to alleviate the problems of data sparsity and cold starts in recommendation systems. However, most of the methods ignore the exploration of feature differentiation aspects in the knowledge propagation process. To solve the above problem, we propose a new attention recommendation method based on an enhanced knowledge propagation perception. Specifically, to capture user preferences in a fine-grained manner in a knowledge graph, an asymmetric semantic attention mechanism is adopted. It identifies the influence of propagation neighbors on user preferences through a more precise representation of the preference semantics for head and tail entities. Furthermore, in consideration of the memory and generalization of different propagation depth features and adaptively adjusting the propagation weights, a new propagation feature exploration framework is designed. The performance of the proposed model is validated by two real-world datasets. The baseline model averagely increases by 9.65% and 9.15% for the Area Under Curve (AUC) and Accuracy (ACC) indicators, which proves the effectiveness of the model.

**Keywords:** recommender systems; knowledge graph; attention mechanism; heterogeneous propagation



**Citation:** Zhang, H.; Wang, Y.; Chen, C.; Liu, R.; Zhou, S.; Gao, T.

Enhancing Knowledge of Propagation-Perception-Based Attention Recommender Systems. *Electronics* **2022**, *11*, 547. <https://doi.org/10.3390/electronics11040547>

Academic Editor: Juan M. Corchado

Received: 7 January 2022

Accepted: 8 February 2022

Published: 11 February 2022

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



**Copyright:** © 2022 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/>).

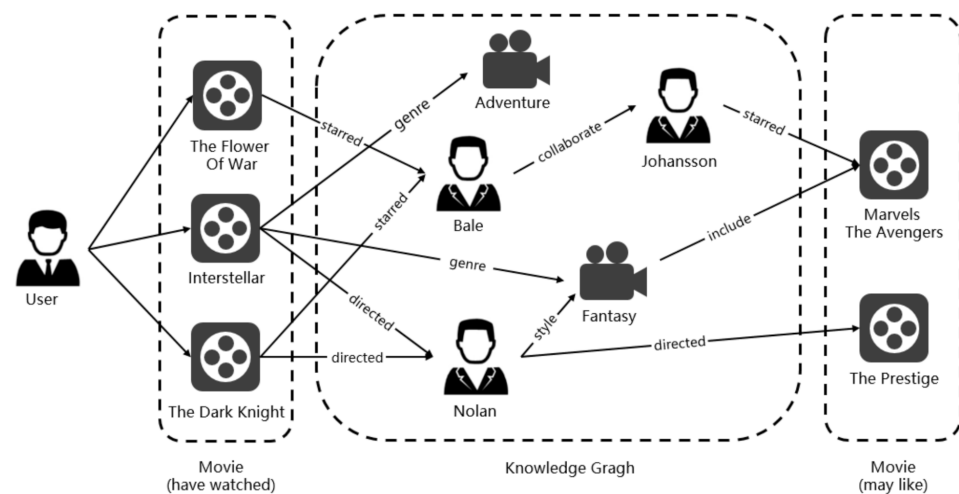
## 1. Introduction

With the rapid growth of data, it becomes very difficult to satisfy the personalized information needs of users. In order to let users find items of interest, recommender systems (RS) were proposed by researchers [1] and applied in electronic business [2], movies [3,4], books and other fields. In recent years, due to the increasing application and better performance on RS in commercial websites such as Amazon and YouTube, it has become a research hotspot.

The collaborative filtering (CF) algorithm is a successful method in the recommendations systems. It adopts the interaction records of users to explore the similarity of users or items, thereby realizing the modeling of user preferences. With the similar users have similar preferences, it recommends the most similar items. CF is mainly divided into two categories [5]: user-based recommendation and item-based recommendation. However, in the application scenario of the Internet, on the one hand, the number of users is much larger than that of items, and the number of resources consumed by users is relatively large. On the other hand, a single user has only a few records in the historical interaction matrix, so it has greater sparseness, which is not conducive to the result of the recommendation. Although item-based recommendations shows the superiority of the algorithm, they can still be affected by data sparseness. In order to solve the above problems and optimize the effect of the recommendation, RS introduces side information auxiliary recommendation,

such as social networks [6], text video content information [7], item reviews [8], etc. Among them, knowledge graphs (KG) are also a kind of efficient side information.

A KG [9] is a large heterogeneous graph composed of triples (head entity, relationship, and tail entity), which can clearly show the relationship among entities in the data. In recent years, many large-scale KGs have appeared continuously, such as DBpedia [10], extracted from Wikipedia and YAGO [11], a knowledge dataset developed by Max Planck Institute in Germany. Because the semantic relationship of the KG alleviates the problems of cold starts and interpretability of the recommender systems [12], enhances the capture of user interest, and then shows strong potential, recommendations based on the KG have attracted the attention of researchers. The recommendation process based on the KG is illustrated in Figure 1. Finding recommended items through the relationships provided by the KG improves the diversity and interpretability of recommendations.



**Figure 1.** Knowledge graph-based recommendation process.

Through recent research, a large number of algorithms combining KG to solve the sparse interactive data have been continuously proposed. For example, DKN [13] is a recommended method based on news that integrates the relevant information of the KG into the news semantics, and then uses a convolutional neural network (CNN) to make the final recommendation. KGCN, proposed by Wang [14] et al., introduced graph neural networks to explore accurate user item embeddings in knowledge graph aggregation. RippleNet [15] uses the corresponding entities of historical items in the KG and then obtains a set of entities with different propagation times by relations. Finally, user preferences are obtained by aggregating the entity features of each hop. It provides a new idea for recommender systems in KG propagation. Later, AKUMP [16] pointed out a different tree-like approach to explore the relationships of entities at each level using a self-attentive mechanism. Recently, Wang et al. [17] combined multi-task learning with RippleNet to improve the recommendation effectiveness of knowledge graphs for outward propagation.

However, most of the existing outward propagation methods ignore the expression of preferences in the semantic features of entities in propagation. This may limit the accuracy of the propagation model for the prediction of items to users, which in turn affects the recommendation gain in the scenario. First, the semantic representation of the head entity to the user's preferences is different under different relationships. Second, only the head entity and the relationship are used to identify the weight of the tail entity, which cannot clearly express the one-to-many relationship of triples. For example, the two triples (Interstellar, genre, and Adventure) and (Interstellar, genre, Fantasy) shown in Figure 1 have the same head entity and relationship. Therefore, it is inappropriate for the tail entity to express the same preference semantics. In terms of feature aggregation at different depths, the existing KG propagation methods only enable a simple summation operation directly on the results obtained from each propagation to obtain deep-level features as user preference

representations. Although the summed higher-order propagated features contain lower-order features, the lower-order features are continuously diluted during the propagation process. The lower-order features are closer to the user's original interaction entity and have a deeper impact on user preferences. Therefore, there are greater restrictions on the expression of preferences.

To solve the above problems, we design a new recommendation method, Enhance Knowledge Propagation Perception Net (EKPNNet). Its purpose is to solve the problem of click-through rate predictions of implicit feedback. First, we initialize the user's click history on the item, and use the dissemination to spread the clicked historical item through the KG to enhance the model's expression of user interest. Then, we use an asymmetrical semantic attention mechanism. When it samples in the KG, the head and tail entities are mapped to the corresponding preference semantic space, thus better expressing the influence of different entities on user preferences. Then, to address the negative impact of propagation depth on the model, we use a new propagation feature exploration architecture that preserves features at different propagation depths and extends the model to a nonlinear neural network to account for feature interactions while taking into account the memory and generalization of features at different depths. Finally, we use a numerical simulation [18] to discuss the effectiveness of EKPNNet. In summary, the contributions of this article are as follows:

1. We design an attention mechanism with asymmetric semantics in EKPNNet for the KG propagation. It enhances the mining of user preferences by mapping the semantics of head and tail entities into different preference spaces.
2. A new communication exploration framework has been proposed. The deep learning network is used to explore the entity characteristics of different depth propagation aggregates in the KG. It balances both the memorization and generalization of features, and adaptively adjusts the weights to different depths.
3. We test our model through a large number of experiments on two real-world public datasets. Compared with several state-of-the-art baselines in multiple indicators, EKPNNet has a substantial improvement.

The structure of this article is as follows: Section 2 introduces research on relevant technologies. Section 3 describes the basic concepts used in the paper and the problems that need to be dealt with. Section 4 shows the model architecture and each module method proposed as well as the loss function, optimization process and time complexity. Section 5 presents the experimental configuration, followed by numerical simulations to make comparisons and a discussion of the results. Finally, conclusions and further work are presented in Section 6.

## 2. Related Work

### 2.1. Knowledge Graph-Based Recommender

The KG is a new area of research for recommender systems, and has been developed rapidly in the recommended industry and academia. At this stage, recommendation methods based on knowledge graphs are divided into three categories: path-based methods, embedding-based methods and unified methods.

The first type is path-based methods. This method was first developed in [19]. Generally, the similarity of the meta path and the path of the user project are compared as the recommendation indicator to enrich the representation of the user or item. For example, Yu et al. [20] designed multiple meta-paths for the first time in the HETE-MF algorithm, then used meta-paths to calculate the similarity among items to obtain a similarity matrix, and finally achieved efficient recommendation through matrix decomposition. Zhao et al. [21] introduced the concept of meta-graph in order to break through the limitation of a single path and then used Factorization Machine (FM) and matrix factorization to further mine the path information. However, these methods must rely on experts with professional knowledge to manually design meta-paths, so many feature combinations

will be missed, and there are countless entity relationships in the context of massive data, which makes it difficult to implement manual design.

The second type is method based on embedding. Usually, this kind of method uses Knowledge Graph Embedding (KGE) [22] to map the entities and relationships into low-dimensional embedding vectors, and then integrate the embeddings into the recommendation framework. For example, Zhang et al. [23] combined three types of data at the same time. It encodes the knowledge graph through the distance model TransR [24], extracts text content and visual content through an auto-encoder, and finally merges the embeddings into the CKE framework. However, it requires a large number of different types of auxiliary information, which is unrealistic for real-world recommendations. Similarly, in [13], a DKN architecture was proposed, which utilized CNN to learn title content, and learned historical clicks through an attention mechanism. Recently, some researchers have applied Graph Neural Networks (GNN) to recommender systems. For example, the BEM framework [25] uses GNN to learn behavior graphs, and uses distance model transE [26] to learn knowledge association graphs, and finally inputs the embedding vectors obtained separately into the Bayesian framework. All in all, although embedding-based methods are widely used in recommendation and show flexibility for multiple data types, this method is more biased towards link prediction problems than recommendation problems.

The third type is unified method. Unified methods combine the advantages of the first two methods. It not only uses the entity semantics of user items but also explores the semantic connectivity of the knowledge graph. Its overall is to use the item embedding of historical interaction and the multi-hop neighbor embedding of item association to make recommendations. For example, RippleNet was proposed by Wang et al. [15]. It provides an efficient end-to-end framework by spreading the historical items of user interaction in the knowledge graph to enhance the expression of user embedding. The two combinations have been proven to be more effective. Later, Tang [16] et al. pointed out a different tree method, AKUPM, which uses the self-attention mechanism to explore the relationship between entities at each level to improve the quality of recommendations. Recently, Wang et al. [17] combined multi-task learning with RippleNet to improve the effectiveness of recommendations. However, the existing methods do not focus on the influence of users' preferences for the semantic features of entities in knowledge propagation, so we design a new architecture for exploring entity preference features in the knowledge propagation.

## 2.2. Attention Mechanism

The attention mechanism simulates the natural reaction of human beings to observe things and to focus on the part of the weight learning to improve the task effect [27]. Nowadays, it has achieved good results in the fields of computer vision [28], waveform analysis [29] and natural language processing [30]. Similarly, it gradually appeared in the field of recommender systems. For example, in [31], the attention mechanism was combined with FM to distinguish the importance of different feature interactions. Zhou et al. [32] proposed DIN in the context of e-commerce, using the attention mechanism to explore the importance of different historical interaction behaviors of users. Wang et al. [13] presented the DKN, which introduced an attention network to calculate the weight of the historical item corresponding to the target item, and then explore the expression of user interest. In [33], a self-attentive integration network framework was presented to capture the interaction among features. Its computational focus was changed from between different sequences to between sequences of itself, and adaptively integrate side information through the integration layer. AKUPM introduces a self-attentive mechanism in the KG propagation, and pays more attention to the relationship between the tail entities in each layer of the propagation. Then, Tu [34] et al. proposed a graph attention method in the field of knowledge graph aggregation. It is the first time that the knowledge distilling and refining method was combined with a graph attention network, and it has obtained better results in knowledge graph aggregation research.



In our work, an attention mechanism with asymmetric semantics is designed for knowledge graph outward propagation, which considers the simultaneous use of head and tail entity preference semantics in the attention mechanism and therefore efficient filtering of irrelevant entities.

### 3. Preliminaries

Before introducing the EKPNet model, it is necessary to clarify the basic concepts and objectives of the task.

#### 3.1. Implicit Feedback

Implicit feedback refers to historical data that do not contain users' tendencies, such as browsing, watching, clicking and other behaviors, which indirectly reflect the users' point of view [35]. Compared with the data form of explicit feedback (such as ratings, etc.), the sparsity of implicit feedback is smaller than that of explicit feedback, and it contains richer information to mine user preferences. In the classic recommendation scenario, the user set is expressed as  $U = \{u_1, u_2, \dots, u_{|U|}\}$  and the item set is expressed as  $O = \{o_1, o_2, \dots, o_{|O|}\}$ . User item pairs are represented by the interaction matrix  $Y = \{y_{uo} | u \in U, o \in O\}$ , where

$$y_{uo} = \begin{cases} 1, & \text{user } u \text{ interacts with item } o; \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

indicates whether there is an interactive relationship between the user and the item. It is also a record of implicit feedback. In particular, an element with a value of 1 in  $Y$  only means that there is an interaction between the user and the item, but it does not mean that the user likes the item. Similarly, the element of 0 in the matrix can only mean that there is no interaction, and it cannot simply say that the user hates it. In this article, we regard the samples with interaction (i.e.,  $y_{uo} = 1$ ) as positive instances, and the samples without interaction (i.e.,  $y_{uo} = 0$ ) as negative samples.

#### 3.2. Knowledge Graph

The other part of the data is the knowledge graph  $G$ . It is composed of the head entity–relationship–tail entity triples  $(h, r, t)$ , which represent the relationship with each entity, and is an efficient data source for the recommender systems. We still use the previous work experience, assuming that each item  $o \in O$  in the interaction matrix  $Y$  has a corresponding entity in the  $G$ .

#### 3.3. Problem Formulation

In this article, our task is to explore the interaction matrix  $Y$  and the knowledge graph  $G$  to predict the probability that users will adopt the target that has not been interacted. More specifically, we have to learn a function

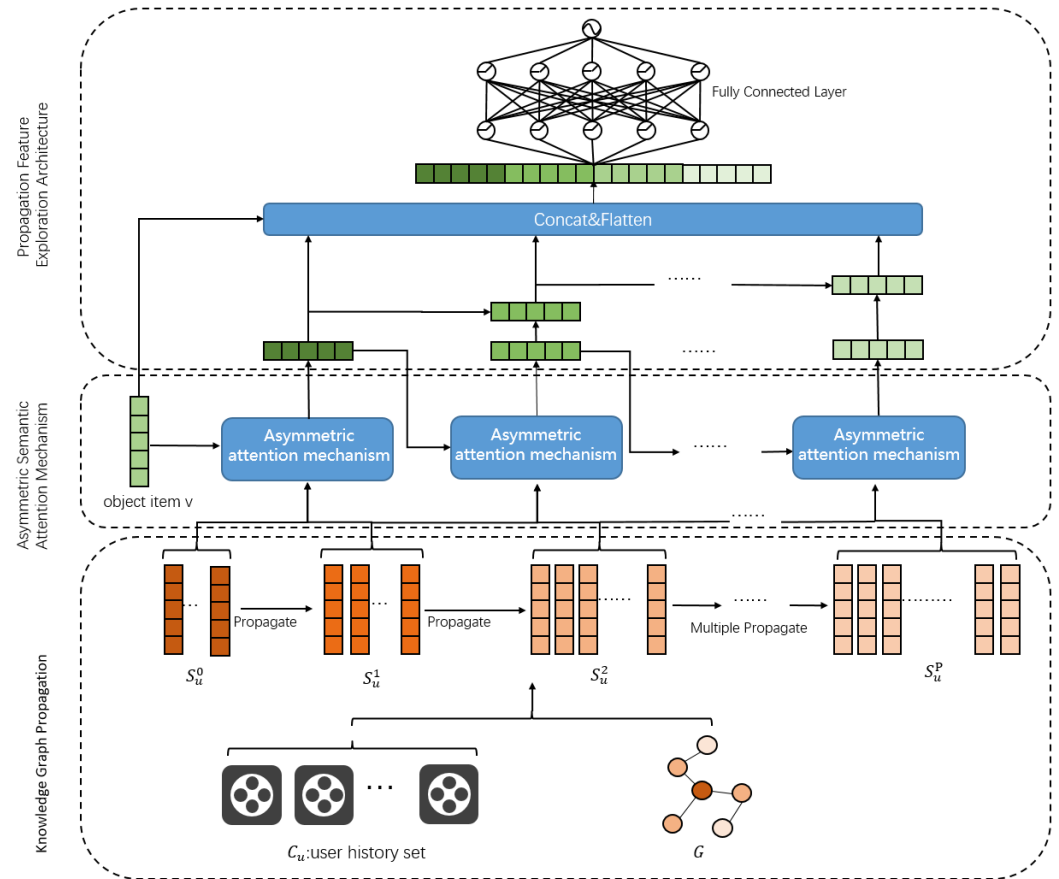
$$\hat{y}_{uo} = F(u, o | \theta, G) \quad (2)$$

where  $\hat{y}_{uo}$  is the predicted probability of interaction between user  $u$  and target item  $o$  obtained by the function,  $\theta$  is the parameter of prediction model  $F$ , and  $G$  is the knowledge graph used for prediction. According to the predicted probability, we can also provide users with a list of recommended top- $N$  results.

## 4. The Proposed Method

The architecture of the EKPNet model is shown in Figure 2. It is composed of three main sections. The first part is the entity propagation layer. It obtains the set of entity representations with different propagation times by the set of user history and the knowledge graph. The second part is the asymmetric attention mechanism layer, which is used for high-quality learning of user preferences in triples. The third part is the propagation

feature exploration prediction layer. Its function is to explore the propagation of each layer returned from the previous layer according to different levels and finally obtain the prediction results.



**Figure 2.** The overall framework of EKPNet.

#### 4.1. Knowledge Graph Propagation

The relationship information among entities is expressed in the knowledge graph. The connected entities have strong relevance, so they have great potential in the exploration of user interests. However, for users, not all entities are useful for expressing user preferences. If entities that are not related to users are introduced into the user preference expression, a large number of data noises will be introduced, which will affect the recommendation result. Therefore, in order to filter the noise of irrelevant entities, we use the knowledge graph propagation method from the user interaction entity as a starting point and use relational connection propagation to explore user-related entities. Assume that all entity sets of knowledge graph  $G$  are  $E$ , and the relation set is  $R$ .

First, for a certain user  $u$ , we retrieve the entity corresponding to the user's historical interaction item in the KG, and obtain the entity set of the user  $u$  at the 0th hop (not yet started to propagation):

$$S_u^0 = \{s_{u,0}^0, s_{u,1}^0, \dots, s_{u,i}^0, \dots, s_{u,n_0}^0\} \quad (3)$$

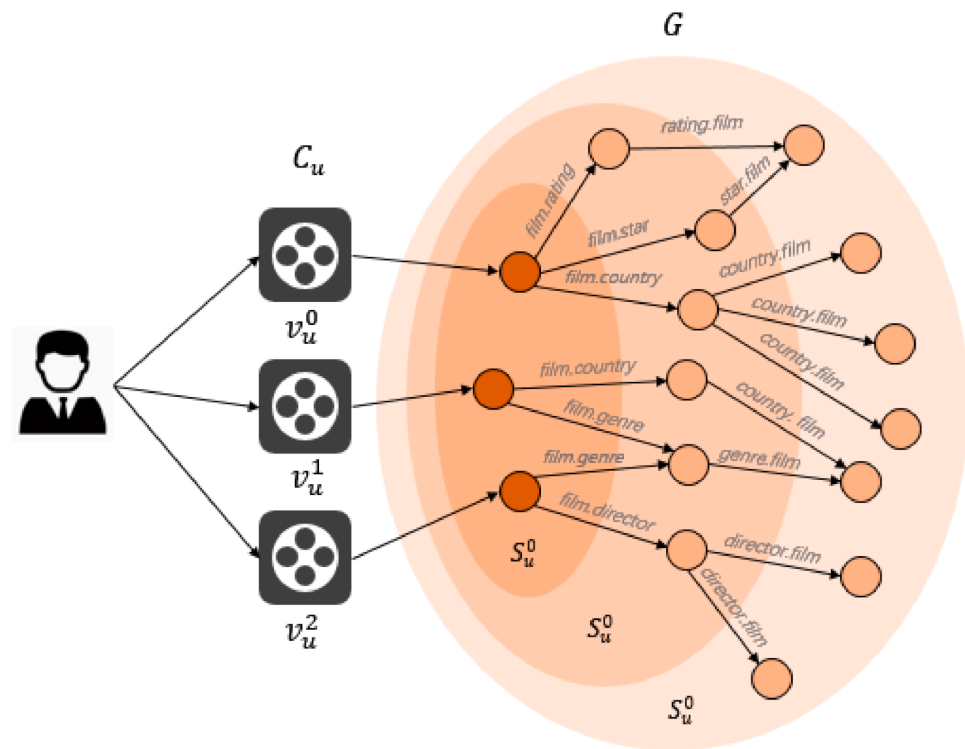
where  $s_{u,i}^0$  is the  $i$ -th entity of the user  $u$  in the 0-th hop set, and  $s_{u,i}^0 \in E = \{e_1, e_2, \dots\}$ . The number in the superscript indicates the current transmission frequency.  $n_0$  is the number of entities in the 0-th hop.

Then, as shown in Figure 3, we use the relationship of the triples as a link to obtain the multi-hop recursive entity representation set of user  $u$ :

$$S_u^p = \left\{ t_u^p \mid \left( h_u^p, r_u^p, t_u^p \right) \in G, h_u^p \in S_u^{p-1} \right\} \quad (4)$$

where  $p = 1, \dots, P$  is the number of propagations. There are  $h_u^p, t_u^p \in E, r_u^p \in R = \{r_1, r_2, \dots\}$ , and if there are repetitive entities in the propagation, they will only remain in the set that appears for the first time. Through the constant iteration of formula (4),  $P + 1$  sets of  $P$  hops of the user are finally obtained:

$$\varepsilon = \left\{ S_u^0, S_u^1, \dots, S_u^P \right\} \quad (5)$$



**Figure 3.** Entity propagation process in the context of movie recommendation.

The above-mentioned knowledge map dissemination process is shown in Figure 3, which clearly shows the process of user history using relationship propagation in the knowledge graph. In the picture, users used three viewing history records to conduct physical propagation twice. Search the three entities corresponding to the historical records in the KG, and use the relationship in the graph to find the associated entities. The entities in the ripples of different color depths represent the set of different propagation times. It is meaningful to use the adjacent entities of the KG to enrich the user representation, because each entity in the graph has a strong relationship with the user, and the adjacent entities can be regarded as the richness of user preferences and item features. Therefore, the propagation process is also a process of exploring the high-level features of the entity.

#### 4.2. Asymmetric Semantic Attention Mechanism

In the  $(h, r, t)$  of the KG, the same head entities have different effects on users' preferences under different relationships. For example, for the entity Forrest Gump movie, the user prefers the genre to the actor; thus, the movie Forrest Gump should have more weight when spreading on the genre relationship. On the one hand, the use of head entities and relationships to identify user preferences cannot identify one-to-many relationships. For

example, the actor relationship in the movie Forrest Gump corresponds to multiple entities. Therefore, it is not appropriate to use the same preference semantic expression for these tail entities. Based on this, we design an asymmetric attention mechanism to map the head and tail entities to the corresponding preference semantic space. Its semantic space is jointly determined by entities and relationships.

In addition, propagation is based on the KG triples. If only the self-attention mechanism of the tail entity is used without the head and tail entity information, the information will be lost. Therefore, in the attention mechanism of this unit, we consider the semantics between the front and back entities. For knowledge graph triples  $(h_i^p, r_i^p, t_i^p)$ , the structure of our asymmetric attention mechanism is as follows:

$$f_{i,o}^p = \varphi\left(\psi\left(v_i^{h_i^p}\right), v_i^{r_i^p}, v_p^o\right) \psi\left(v_i^{t_i^p}\right) \quad (6)$$

where  $f_{i,o}^p$  is the feature vector of the  $i$ -th entity in the  $p$ -th propagation of a certain user  $u$  corresponding to the target item  $o$ , which is the output part of the asymmetric attention mechanism.  $\varphi(\cdot, \cdot, \cdot)$  is the similarity calculation function, which is used to measure the attention weight of the propagating entity.  $\psi(\cdot)$  is a semantic transformation function of asymmetric preference, which enables entities to express preference semantic features in the attention mechanism.  $v_i^{h_i^p}, v_i^{r_i^p}, v_i^{t_i^p} \in \mathbb{R}^d$  are the embedding of the head entity  $h_i^p$ , the relation  $r_i^p$  and the tail entity  $t_i^p$  of the  $i$ -th triplet in the user's  $p$ -th propagation.  $v_p^o \in \mathbb{R}^d$  is the embedding of the target item  $o$  at the  $p$ -th hop. However, the  $v_p^o$  of each hop is constantly updated. At the end of this section, we give the update method of  $v_p^o$ . Below we will introduce several functions in the above formula one by one. We first introduce the semantic transformation function  $\psi(\cdot)$  of asymmetric entities. In the triple  $(h_i^p, r_i^p, t_i^p)$ , the corresponding head-to-tail entity semantic transformation formula is as follows:

$$\begin{aligned} \psi\left(v_i^{h_i^p}\right) &= T_{\langle h_i^p, r_i^p \rangle} v_i^{h_i^p} \\ \psi\left(v_i^{t_i^p}\right) &= T_{\langle r_i^p, t_i^p \rangle} v_i^{t_i^p} \end{aligned} \quad (7)$$

where  $T_{\langle h_i^p, r_i^p \rangle} \in \mathbb{R}^{d \times d}$  and  $T_{\langle r_i^p, t_i^p \rangle} \in \mathbb{R}^{d \times d}$  are the semantic matrix corresponding to the head and tail entities. The head and tail entities are mapped to different semantic spaces through their respective semantic matrices to realize the expression of their respective preference semantics. Although the use of different semantic matrices can enhance the expression of semantic features, on the one hand, the number of entities in the KG is a lot, and for each different combination of head entity and relationship or tail entity and relationship, there are different semantic matrices; these will result in a sharp increase in the number of parameters. On the other hand, if a certain head entity relationship combination or tail entity relationship combination does not include in the training set, the matrix will not be trained, resulting in a sharp drop in recommendation performance.

Therefore, for the above-mentioned problem of our model, we used a similar method to the translation model TransD [36] to optimize the transfer function. However, unlike TransD, our focus is on optimizing perception in the attention mechanism rather than translating entities. Specifically, first construct a semantic vector  $\gamma \in \mathbb{R}^d$  for each entity and relationship, and then take out the three semantic vectors  $\gamma_h, \gamma_r, \gamma_t$  corresponding to the head and tail entities and the relationship in the triples. Finally, the semantic vectors are, respectively, outer product with the relational semantic vector to obtain the semantic matrix of the head and tail entities. Then, the above formula, Formula (7), becomes the following formula:

$$\begin{aligned} \psi\left(v_i^{h_i^p}\right) &= (\gamma_{h_i^p} \otimes \gamma_{r_i^p}^T) v_i^{h_i^p} \\ \psi\left(v_i^{t_i^p}\right) &= (\gamma_{t_i^p} \otimes \gamma_{r_i^p}^T) v_i^{t_i^p} \end{aligned} \quad (8)$$

where  $\gamma_{h_i^p}$  and  $\gamma_{t_i^p}$  are the corresponding semantic vectors of the  $h_i^p$  and  $t_i^p$ .  $\gamma_{r_i^p}$  represents the semantic vector of the corresponding relationship between the head and tail entities, and  $\gamma \in \mathbb{R}^d$ .  $\otimes$  denotes the vector outer product operation.

Before calculating the similarity calculation function  $\phi(\cdot, \cdot, \cdot)$ , we first introduce a triplet with a target item association weight function  $\phi(\cdot, \cdot, \cdot)$ . It maps the degree of matching between the triple and the target item by the form of the dot product similarity of  $h + r$  with the target item. Therefore, the association weight is expressed as:

$$\phi(\psi(v^{h_i^p}), v^{r_i^p}) = (\psi(v^{h_i^p}) + v^{r_i^p})v_p^o \quad (9)$$

Finally, we use the softmax function to integrate the results of Formula (9) to obtain the weight of the importance of each triplet in a single propagation:

$$\varphi\left(\psi\left(v^{h_i^p}\right), v^{r_i^p}, v_p^o\right) = \frac{\exp(\phi(\psi(v^{h_i^p}), v^{r_i^p}))}{\sum_{i=1}^n \exp(\phi(\psi(v^{h_i^p}), v^{r_i^p}))} \quad (10)$$

By using the above equation, we obtain the triad attention weights, and the weighted attention mechanism allows us to understand which triad should be given more attention. The tail entities are the subjects after propagation; thus, a single weighted tail entity feature vector  $f_{i,o}^p$  is obtained by multiplying the tail entities with their corresponding attention weights by using Equation (6). Finally, the weighted results of all entities of a whole hop are summed to obtain a representation of the feature embedding of the  $p$ -th hop:

$$\rho_{u,o}^p = \sum_{i=1}^{n_p} f_{i,o}^p \quad (11)$$

where  $n_p$  is the number of  $p$ -th hop tail entity. After going through the asymmetric semantic attention mechanism, we obtain the characteristics of a whole hop. For after each propagation, the next hop is associated with the previous hop entity. Therefore, the representation of the items that is used for an association comparison needs to be updated. So, a single-layer neural network is used to update the representation vector. The specific form is as follows:

$$v_p^o = \left(v_{p-1}^o + \rho_{u,o}^p\right)W_e + b_e \quad (12)$$

where  $W_e \in \mathbb{R}^{d \times d}$  and  $b_e \in \mathbb{R}^d$  are the weight matrix and bias vector during transformation. Finally, we summarize the feature of all the hops, and obtain the set of all the features of the  $p$  hops of the item  $o$  in the  $G$  by the user  $u$ :

$$\tau_{u,o} = \left\{\rho_{u,o}^1, \rho_{u,o}^2, \dots, \rho_{u,o}^p\right\} \quad (13)$$

#### 4.3. Propagation Feature Exploration Architecture

Through the asymmetric attention mechanism, we obtain the representation vector of the feature of multi-hops propagation. The representation vector of each layer can be understood as the preference feature of different times of propagation. Based on past experiences, previous propagation models use a simple addition vector aggregation method. This method first obtains the user's representation vector by summing the feature vectors propagated by the user  $p$  times and then outputs the prediction result by a dot product operation with the representation vector of item  $o$ . Finally, the prediction result is output by the sigmoid function. The calculation process of this method is as follows:

$$v^u = \xi_{u,o}^p = \rho_{u,o}^1 + \rho_{u,o}^2 + \dots + \rho_{u,o}^p \quad (14)$$

$$\hat{y}_{uv} = \sigma\left(v^{uT}v^o\right) \quad (15)$$



where  $v^u$  is the embedding of  $u$ , and  $v^o$  is the original feature embedding of  $o$ .  $\sigma(x) = \frac{1}{1 + \exp(x)}$  is the sigmoid function. The direct addition method exploits user interaction history items and item propagation characteristics to represent user features. The advantage of this method is that it has a good expression and feature constraint effect for users' interest feature, and it converges quickly. However, its shortcomings are also obvious. Each addition operation will dilute the low-level features of the propagation, and the low-level features need higher weights because they are closer to the user's original features. In addition, it ignores the interactive exploration between features. Therefore, we propose a new propagation feature exploration architecture to make full use of its advantages and solve its shortcomings.

First, we pre-train the weights through Formulas (14) and (15) to obtain the embedding vector and attention semantic vector of each entity and relationship in this model. On the one hand, because in the propagation process, the feature with the smaller number of propagation times is closer to users' original feature, and the more important it is to express the users' feature. Therefore, the proportion of features with a smaller number of propagation times should be increased. On the other hand, the aggregated user vectors of different propagation times all express the characteristics of users with different propagation depths. Therefore, we use the idea of Wide&Deep [37] to keep the feature vectors of low and high order at the same time to ensure the memory and generalization of features. The model summarizes  $P$  user feature vectors of different depths  $\xi_{u,o}^p$  through formula (14), and each order feature includes the feature of the previous propagation order to enhance the weight of the original feature. These features plus the feature vector of the target item  $o$  make the  $P + 1$  feature vectors perform linking operations. Additionally, in order to explore the deep interaction among features,  $n$ -layer neural network is used to process the connected vector, and finally obtain the prediction result. The details are as follows:

$$z_0 = \xi_{u,o}^1 || \xi_{u,o}^2 || \dots || \xi_{u,o}^P || v_p^o \quad (16)$$

$$z_l = a(z_{l-1}W^l + b^l), l = 1, 2, \dots, L \quad (17)$$

$$\hat{y}_{uo} = \sigma(z_{L-1}W^L + b^L) \quad (18)$$

where  $W^l$  and  $b^l$  are the weights and biases in the neural network and  $||$  is a vector connection.  $l$  indicates which layer is in the neural network, and  $L$  is the total number of network layers.  $z_0$  is the input of the neural network and  $z_l$  is the output of the  $l$ th neural network layer.  $a(\cdot)$  represents the activation function of the neural network. In the experiment, we choose different nonlinear activation functions for the hidden layer and compare them. Finally, we obtain the final probability result through sigmoid.

#### 4.4. Learning Algorithm

In this article, we use the given knowledge graph  $G$  and the user-item interaction matrix  $Y$  to calculate the maximum posterior probability of the model parameter  $\Theta$ .  $\Theta$  contains all the parameters in the model, such as the representation vectors of head and tail entities and relationships  $v^h, v^t, v^r$ , the corresponding semantic vectors  $\gamma_h, \gamma_t, \gamma_r$  in the attention mechanism, the parameters  $W_e$  and  $b_e$  for item transformation in the propagation process, and the parameters  $W^l$  and  $b^l$  of the neural network in the deep exploration framework. The form of the posterior probability of maximizing  $\Theta$  is as follows:

$$\operatorname{argmax}(p(\Theta|G, Y)) \quad (19)$$

The maximization formula is equivalent to maximizing the following formula, and is obtained by reasoning through Bayes' theorem:

$$\begin{aligned} p(\Theta|G, Y) &= \frac{p(\Theta, G, Y)}{p(G, Y)} \propto p(\Theta, G, Y) \\ &= p(\Theta)p(G|\Theta)p(Y|\Theta, G) \\ &= p(\Theta)p(G|\Theta)p(Y|\Theta) \end{aligned} \quad (20)$$

where  $p(\Theta)$  is the prior probability of the model parameter  $\Theta$ ,  $p(G|\Theta)$  is the posterior probability of the knowledge graph  $G$  under the premise of the given parameter  $\Theta$ ,  $p(Y|\Theta)$  is the premise of the given parameter  $\Theta$ . Observe the posterior probability of interaction  $Y$ . In the above-mentioned maximization problems, according to the rules of Maximum Likelihood Estimation (MLE), the logarithm of the original expression is taken and transformed into a multinomial sum. Then, take the opposite number of the function, transform the objective optimization problem into a minimization optimization problem, and obtain the objective function  $E$ , as shown below:

$$E = -\log(p(\Theta|G, Y)) = -\log p(G|\Theta) - \log p(Y|\Theta) - \log p(\Theta) \quad (21)$$

The function in the first term of Formula (22) shows the accuracy of the model's coding accuracy of the entity relationships. We use TransE's knowledge graph coding method, assuming that it conforms to the Gaussian distribution and defines the following likelihood function:

$$\begin{aligned} \log p(G|\Theta) &= \log \prod_{(h,r,t) \in G} \mathcal{N}(I^{h,r,t} - \sigma(v^h + v^r - v^t), \lambda_1) \\ &= -\frac{\lambda_1}{2} \sum_{(h,r,t) \in G} \|I^{h,r,t} - \sigma(v^h + v^r - v^t)\|_2^2 \end{aligned} \quad (22)$$

where  $I^{h,r,t}$  indicates whether this triplet exists in graph  $G$ . If it exists, it is 1; otherwise, it is 0. For the second term in Formula (22), for each interaction  $y_{uo} \in Y$ , The likelihood function can be constructed using the Bernoulli distribution

$$\begin{aligned} \log p(Y|\Theta) &= \log \prod_{u,v \in Y} \hat{y}_{uv}^{y_{uv}} \cdot (1 - \hat{y}_{uv})^{1-y_{uv}} \\ &= \sum_{y_{uv} \in Y} y_{uv} \log \hat{y}_{uv} + (1 - y_{uv}) \log(1 - \hat{y}_{uv}). \end{aligned} \quad (23)$$

The last term  $p(\Theta)$  in (22) is a priori probability that can be ignored in extreme value problems. Finally, by substituting Formulas (23) and (24) into (22), and adding the l2 regularization of the parameters, we obtain the final loss function:

$$\begin{aligned} L &= - \sum_{u,v \in Y} y_{uv} \log \hat{y}_{uv} + (1 - y_{uv}) \log(1 - \hat{y}_{uv}) \\ &+ \frac{\lambda_1}{2} \sum_{(h,r,t) \in G} \|I^{h,r,t} - \sigma(e^h + e^r - e^t)\|_2^2 + \frac{\lambda_2}{2} (\|w\|_2^2 + \|b\|_2^2) \\ &+ \frac{\lambda_3}{2} (\|h\|_2^2 + \|r\|_2^2 + \|t\|_2^2 + \|\gamma_h\|_2^2 + \|\gamma_r\|_2^2 + \|\gamma_t\|_2^2) \end{aligned} \quad (24)$$

For the above loss function, we use the advanced Adam [38] optimization method to solve the local optimal problem. We give parameter pre-training and parameter optimization processes of model training in Algorithm 1 and Algorithm 2, respectively.

Time complexity: Algorithm 1 is a pre-trained model of propagation with time complexity  $O(\sum_{i=0}^p n * k^2)$ , where  $n$  is the size of the fixed sampling triplet in propagation and  $k$  is the dimension of the transformation matrix. Algorithm 2 adds the

propagation feature exploration layer based on Algorithm 1 so the time complexity is  $O\left(\sum_{i=0}^p n * k^2 + \sum_{i=0}^l k_{l-1} k_l\right)$ , where  $k_l$  and  $k_{l-1}$  represent the current and previous transformation sizes.

---

**Algorithm 1** Learning in the pre-training for EKPNet

---

**Input:** implicit feedback matrix  $Y$ , knowledge graph  $G$

**Set:** batch size  $b$ , learning rate  $\alpha$ , dimensionality  $k$

1. Initialize  $v^e, v^r$  according to TransE Algorithm and randomly initialize
2.  $\gamma_e, \gamma_r, W_e, b_e, \beta_1 = 0.9, \beta_2 = 0.999, \eta = 0.002$
3. Retrieve the set  $S_u^p$  for each user  $u$  from the knowledge graph  $G$
4. **for** number of training sample **do**
5. Sample a mini batch of true and false triples from  $G$ ;
6. Sample mini batch of positive and negative interactions from  $Y$ ;
7. Prediction value  $\hat{y}_{uo}$  via Equations (6)–(15)
8. Calculate the loss and gradients  $\frac{\partial L}{\partial \Theta}$  of all parameters  $\Theta$  via Equations (19)–(24);
9. Initialize  $v_0, s_t, t \leftarrow 0$
10. **while**  $\theta_t$  not converged **do**
11.  $v_t \leftarrow \beta_1 v_{t-1} + (1 - \beta_1) \frac{\partial L}{\partial \Theta}$
12.  $s_t \leftarrow \beta_2 s_{t-1} + (1 - \beta_2) \frac{\partial L}{\partial \Theta} \circ \frac{\partial L}{\partial \Theta}$
13.  $\Theta_{t+1} \leftarrow \Theta_t - \eta \frac{v_t}{\sqrt{s_t}}$  (update parameters)
14. **end while**
15. **end for**
16. **Return**  $v^e, v^r, \gamma_e, \gamma_r, W_e, b_e$

**Output:** Parameters of the model  $v^e, v^r, \gamma_e, \gamma_r, W_e, b_e$

---



---

**Algorithm 2** EKPNet Learning Algorithm

---

**Input:** implicit feedback matrix  $Y$ , knowledge graph  $G$ , The parameters of Algorithm 1.

**Set:** batch size  $b$ , learning rate  $\alpha$ , dimensionality  $k$

1. Initialize  $e^E, e^R, \gamma_E, \gamma_R, W_e, b_e$  according to TransE Algorithm and randomly initialize  $W_i, b_i$
2.  $\beta_1 = 0.9, \beta_2 = 0.999, \eta = 0.0001$
3. Retrieve the set  $S_u^p$  for each user  $u$  from the knowledge graph  $G$
4. **Repeat**
5. **for** number of training sample **do**
6. Sample a mini batch of true and false triples from  $G$ ;
7. Sample mini batch of positive and negative interactions from  $Y$ ;
8. Prediction value  $\hat{y}_{uo}$  via Equations (6)–(13) and Equations (16)–(18)
9. Calculate the loss and gradients  $\frac{\partial L}{\partial \Theta}$  of all parameters  $\Theta$  via Equations (19)–(24);
10. The parameters are updated in the same way as in Algorithm 1
11. **end for**
12. **Return**  $v^e, v^r, \gamma_e, \gamma_r, W_e, b_e$

**Output:** results predictions

---

## 5. Experimental Results and Discussion

### 5.1. Experiment Preparation

#### 5.1.1. Dataset

We choose two real world datasets in completely different backgrounds to prove the effectiveness of the model in different scenarios. The datasets are as follows:

MovieLens-1M [39] is a movie rating dataset widely used in the field of movie recommendations. It is the result of the GroupLens research project by the University of Minnesota. It includes 1,000,209 score records of 6040 users with a value of 1–5 for 3900 movies. It also contains basic movies and user attributes, such as age, gender, occupation, and movie category.

Book-Crossing [40] is a book scoring dataset widely used in the field of book recommendation. The data are collected in the Book-Crossing community. It contains 1,149,780 scoring records of 271,379 books by 278,858 users with a value of 1–10. It contains basic demographic data and basic information about the book.

Since the goal of our experiment is to predict the probability of interaction between the target user and the item through implicit feedback, the scoring data must be converted into implicit data. Specifically, we set a threshold in the dataset to define whether the user interacts. For each record, we set the score to be not lower than the threshold as positive feedback data (label 1) and randomly sample the data of non-interactive user items as negative feedback data (label 0), and the sampling size is the same as the number of positive samples. In addition, we will delete users who do not include positive implicit feedback in the experiment, because the positive implicit feedback of users is needed to verify the results in model testing.

The knowledge graph used in this article was published in [15]. The methods of constructing a knowledge graph for movie datasets and book datasets are the same. First, a subset is constructed by matching the movie name and book name entries in the dataset with the items in the Microsoft Satori knowledge base. Then, delete the unmatched item or multiple matched items, and finally, given the matched knowledge graph, the entity set is expanded to four hops. Through the above process, MovieLens finally obtained 1,241,995 triples corresponding to 182,011 entities and 12 kinds of relationships, and Book-Crossing finally obtained 77,903 entities and 198,771 triples corresponding to 25 kinds of relationships. In order to express this more clearly, the statistics of the two datasets are shown in Table 1, where inter-avg indicates the average interaction of each item, link-avg indicates the average number of links per item, and sparsity indicates the sparsity of the dataset.

**Table 1.** Statistics of two datasets: MovieLens-1M and Book-Crossing.

		MovieLens-1M	Book-Crossing
User–Item Interaction	#users	6036	17,860
	#items	2445	14,967
	#interaction	753,772	139,746
	#inter-avg	125	8
	# sparsity	94.89%	99.94%
Knowledge Graph	#entities	182,011	77,903
	#relations	12	25
	#KG triples	1,241,995	198,771
	#link-avg	7	3

### 5.1.2. Baseline

Six advanced recommendation models are included in the experiment to verify the effectiveness of EKPNet. After reproducing the following model, we set the baseline parameters to the best parameters in the original paper.

CKE [23] is a classic embedding-based recommendation model that integrates knowledge graph, vision, and text into the Bayesian recommendation framework. However, since the dataset does not contain image and text information, we simplify the model and use TransR encoding, with the embedding used as the model input.

DKN [13] is a news recommendation framework based on embedding. It uses word and entity embedding as the model input to construct the feature vector of each news, and then explores the user's interest through the attention mechanism formed by CNN. In this experiment, we use movie titles and book titles as the model text input.

Wide&Deep [37] is one of the most classic general recommendation frameworks in the recommender system. The input features are processed through the linear part of wide and the nonlinear part of deep to consider the depth and shallowness of feature exploration at

the same time. The ID of the item and the embedding obtained after the knowledge graph TransR are used as the input of the model.

RippleNet [15] is a most advanced unified high-efficiency model that combines the advantages of path-based and embedding-based methods. It uses user click history to propagate potential preferences to obtain user representations, and finally combines with item representations to predict the interaction probability.

KGCN [14] is another state-of-the-art kind of unified approach. It enables non-spectral graph convolutional network methods to be applied to knowledge graph to efficiently learn knowledge graph information as well as to mine potential features of users.

Ripp-MKR [17] is an improved method based on RippleNet, which integrates multi-task learning into the communication framework, and learns the knowledge graph and historical interaction matrix at the same time.

### 5.1.3. Experimental Settings

In the EKPNet model experiment, for two datasets, the ratio of our training, validation and test set is set to 6:2:2. We conducted three experiments for each result and took the means. To demonstrate the validity of the model, the evaluation indicators were selected through two tasks: CTR click prediction evaluation and TOP-K recommendation. For the CTR task, we chose AUC and ACC indicators. For TOP-K recommendation evaluation, we chose Precision@K, Recall@K and F1@K evaluation indicators, where K is the number of recommended items. In order to show the recommendation effect in different K, K was set to 1, 2, 5, 10, 20, 50, 100.

The specific settings of pre-training parameters and training parameters are shown in Table 2, where  $d$  represents the feature embedding dimension,  $P$  represents the maximum number of propagations,  $N$  represents the number of propagation-selected entities,  $\lambda_1$ ,  $\lambda_2$ ,  $\lambda_3$  represent the regularization coefficients, and  $L$  represents the number of neural network layers.

**Table 2.** The detailed parameters of the EKPNet experiment.

	$d$	$P$	$N$	$\lambda_1$	$\lambda_2$	$\lambda_3$	$lr$	$L$
Movielens-pre	16	2	32	0.01	$5 \times 10^{-3}$	-	$1 \times 10^{-2}$	-
Movielens	16	2	32	0.001	$1 \times 10^{-7}$	$1 \times 10^{-7}$	$5 \times 10^{-3}$	3
BookCrossing-pre	4	3	32	0.01	$1 \times 10^{-7}$	-	$1 \times 10^{-2}$	-
BookCrossing	4	3	32	0.001	$1 \times 10^{-4}$	$1 \times 10^{-4}$	$5 \times 10^{-3}$	4

The experimental hardware environment includes 64-core Intel(R) Xeon(R) CPU, A100-PCIE-40GB, memory 48G, and system Ubuntu. The software uses Python programming language, PyTorch framework and NumPy scientific computing library.

### 5.2. Experimental Results

Table 3 and Figures 4–6 show the performance of the model in CTR prediction and TOP-K recommendation tasks, respectively. Through them, we can find that:

1. Compared with these state-of-the-art baselines, EKPNet has a better effect on datasets in two different fields. More specifically, the asymmetric semantic attention mechanism and the multi-level propagation feature exploration architecture play an active role in mining user preferences and providing recommended items. Compared with the effect of baseline on the MovieLens-1M and Book-Crossing datasets, EKPNet has average increased AUC values by 10.8% and 8.5%, and ACC by 11.2% and 7.1%.
2. For the top-K task our model has a greater advantage on the MovieLens-1M dataset. Its relatively low sparsity in KG may lead to better training of the semantic vectors we introduced. Another reason why the EKPNet model improves less in the Book-Crossing dataset compared to Ripp-MKR is that Ripp-MKR introduces neural collaborative filtering, while the Book-Crossing dataset has more interaction infor-

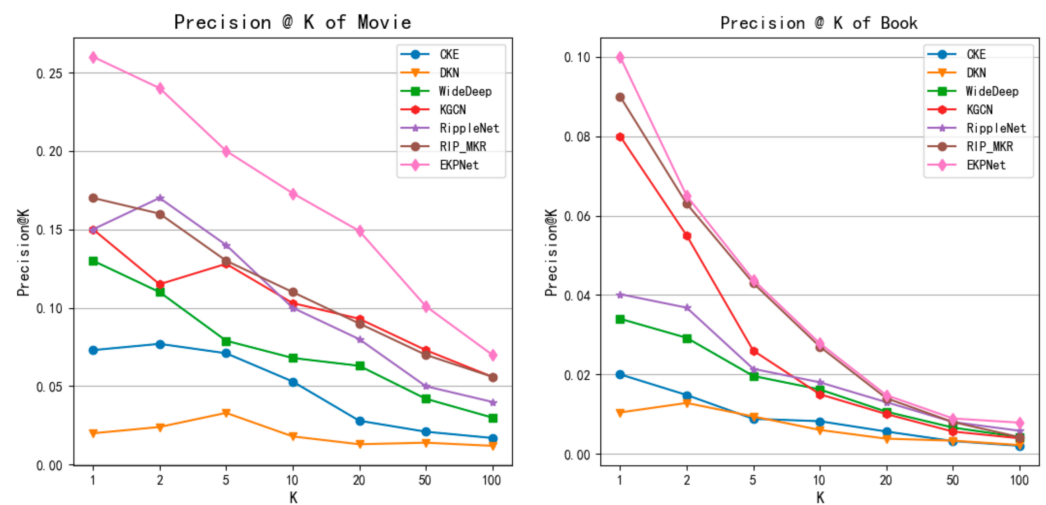
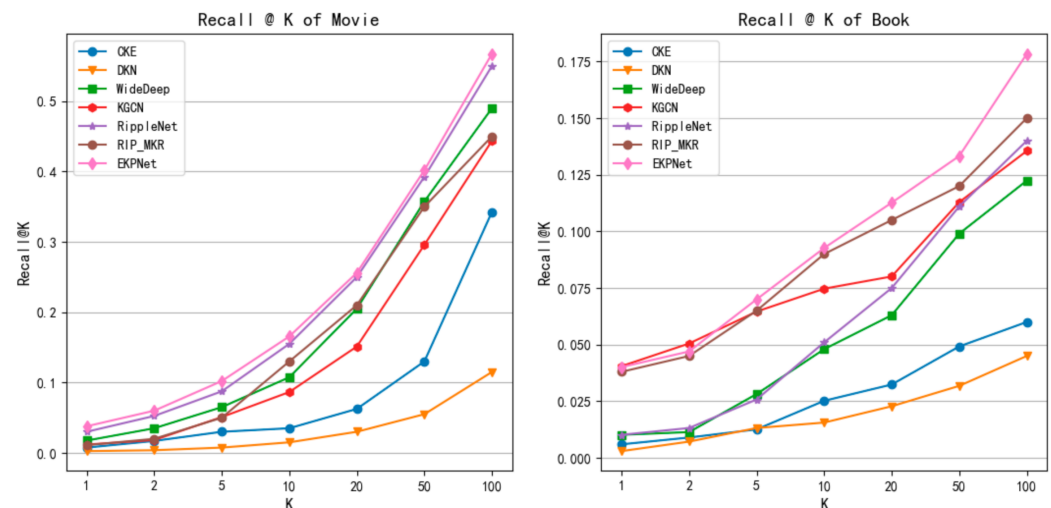


mation to assist joint learning. In addition, our model targets the CTR task and the cross-entropy loss used in the loss function, rather than the selection ranking loss, so there are some limitations for the ranking task.

3. By observing the results between the two data, it is found that the results of the Book-Crossing dataset in all models are lower than the MovieLens-1M dataset. The main reason is shown in Table 3 in Section 5.1. On the one hand, in the user project interaction matrix, the Book-Crossing dataset has an average number of 117 fewer interactions per user than the MovieLens-1M dataset, with a difference of 5.05% in sparsity. On the other hand, in the knowledge graph dataset of the two data, the Book-Crossing dataset has an average of four fewer links per entity than the MovieLens-1M dataset. The sparse data make each model have insufficient information to explore the characteristics of user items and its performance is not satisfactory.
4. Compared with other baselines, DKN's method is overwhelmed by most models, because compared with news titles, the names of movies and books are shorter, the information contained is very limited, and it is difficult to reflect the characteristics of movies or books.
5. The poor performance of the CKE model for us may be due to the lack of image and text information in the data, which limits the expression of the model. Then, unlike RippleNet and EKPNet, the model only directly uses related entities to represent features, and does not use related entities to enrich feature representations.
6. Compared with the above two baselines, Deep&Wide achieved relatively good results, indicating that deep learning obtains better results in exploring the intersection of entity features. However, compared with the other three unification-based methods the results are poor, and it can be seen that exploring the knowledge graph is crucial for the prediction results.
7. RippleNet, KGCN and Ripp-MKR exhibit the best performances among all the baselines. Therefore, it is proved that exploring structural information in the knowledge graph can effectively improve the model effect.
8. EKPNet has a clear advantage over RippleNet and Ripp-MKR in both tasks. It demonstrates that in outward propagation, considering the impact of distinguishing the semantics of different classes of head and tail entities in the attention mechanism and exploring the aggregation features at different levels of propagation have a positive impact on the final results. The EKPNet also exhibits a better performance compared to KGCN, proving that our model also has some advantages compared to graph convolutional networks.
9. For the inward aggregated attention mechanism model KCAN, we obtained an AUC of 0.907 in the MovieLens-1M dataset and 0.928 in the EKPNet model. Our model improved by 2.31% compared to KCAN. However, KCAN treats users as entity vectors in the knowledge graph, and every time a new user interaction occurs, the graph needs to be reconstructed and the entire model retrained. Additionally, our work uses the user's interaction history items in the knowledge graph outward propagation to indicate that the user can solve the problem of new users appearing. Therefore, EKPNet exhibits a better performance.

**Table 3.** The performance of different baselines in the two datasets in the CRT prediction task.

Model	MovieLens-1M			Book-Crossing				
	AUC	imp	ACC	imp	AUC	imp	ACC	imp
CKE	0.765	21.31%	0.719	19.05%	0.654	13.30%	0.625	8.32%
DKN	0.683	35.87%	0.612	39.87%	0.631	17.43%	0.604	12.09%
Wide&Deep	0.896	3.57%	0.823	4.01%	0.693	6.93%	0.621	9.02%
RippleNet	0.917	1.20%	0.844	1.42%	0.698	6.16%	0.640	5.78%
KGCN	0.907	2.1%	0.838	1.8%	0.687	5.4%	0.631	4.6%
Ripp-MKR	0.920	0.8%	0.845	1.1%	0.720	2.1%	0.650	2.7%
EKPNet	0.928	-	0.856	-	0.741	-	0.677	-

**Figure 4.** Precision@K of different baselines in the two datasets in the topK task.**Figure 5.** Recall@K of different baselines in the two datasets in the topK task.

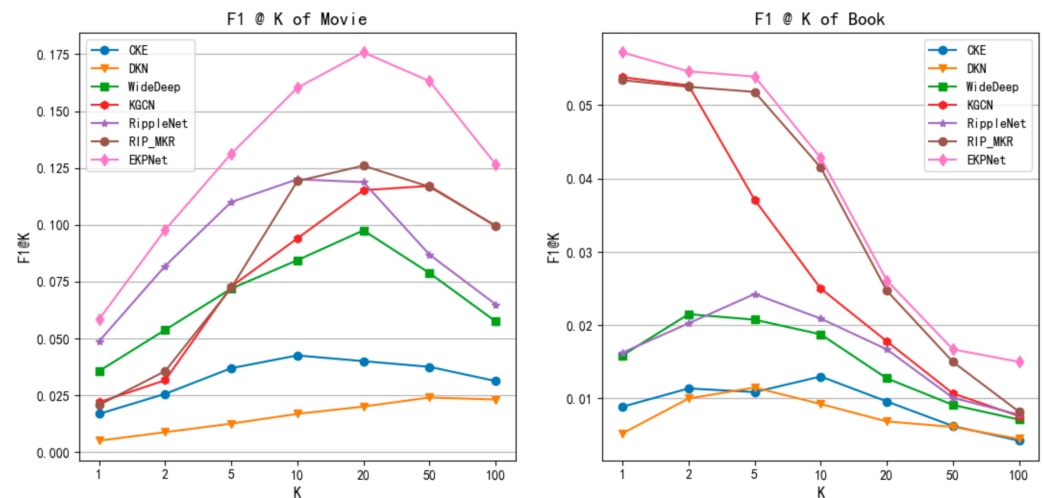


Figure 6. F1@K of different baselines in the two datasets in the topK task.

### 5.3. The Influence of Activation Function and MLP Structure

In the deep exploration framework, choosing the structure of the deep model and the activation function of each layer can directly affect the effect of the model. To verify the relationship between the model and MLP, we designed a comparative experiment. The two datasets in the above experiment were used to observe the performance of different layers and different activation functions (such as relu function, sigmoid function, and tanh function).

The results are shown in Figure 7. For the activation function, since previous studies have shown that the two datasets are quite different in either the knowledge graph or the interaction matrix, the model performance is quite different. For the MovieLens-1M dataset, relu and tanh show similar performances, but for the Book-Crossing dataset, the effect of the tanh function is significantly better than relu, which may be due to the normalization ability of tanh in line with the characteristic distribution of the data. The same is that the performance of the sigmoid function is always the worst.

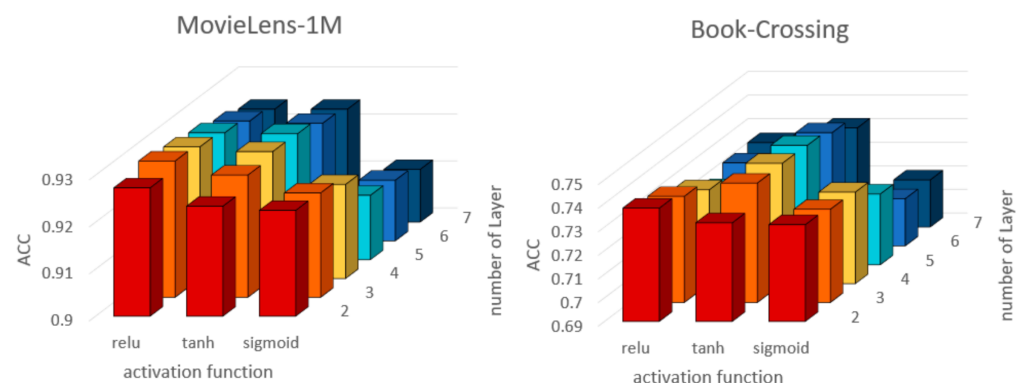


Figure 7. Model performance under different activation functions and different quantity levels.

For the MLP structure related to the capacity of the model, we used the same width as the input, and chose six different depths of mlp to explore the effect of depth. In the two datasets, the relu and sigmoid functions are more sensitive to depth, while tanh is relatively stable. The experimental results show that it has the best performance when the structure is made up of four layers. A shallower model cannot perfectly fit the data features, and the problem of gradient disappearance leads to a deeper model structure that will make the effect worse. Therefore, a deeper MLP cannot ensure a higher effect.

Finally, we conclude that the overall effect of the model is the best when the four-layer mlp and tanh activation functions are selected.

#### 5.4. Module Ablation Experiment

In order to explore the necessity of the proposed two parts for the model effect, we compare the original model with three different model variants. EKP\_no represents the simplest propagation model, EKP\_att represents only the asymmetric semantic attention mechanism, and EKP\_frame is a model that only uses the propagation feature to explore the architecture. EKPNet represents a complete model that uses two modules at the same time. The final result is shown in Table 4. Using the attention mechanism and the propagation feature exploration framework alone in the MovieLens-1M dataset boosted the effect by 0.45% and 0.51%, respectively. Using both modules together boosted it by 1.22%. Using the two parts alone in the Book-Crossing dataset improves it by 1.41% and 3.50%, respectively, and using both modules at the same time improved it by 5.73%. EKP\_att and EKP\_frame both saw a large improvement compared to the original propagation model, indicating the effectiveness of the two improved parts in the propagation model. Additionally, the two parts have more similar enhancement effects for MovieLens-1M model. However, EKP\_frame has better results in the Book-Crossing dataset, probably because the Book-Crossing data are sparse and the vector of the attention mechanism is not fully trained. The effect also shows that the combination of the two parts yields better results.

**Table 4.** The performance of different variants in the CRT prediction scenario.

ACC	EKP_no	EKP_att	EKP_frame	EKPNet
MovieLens-1M	0.9172	0.9213	0.9219	0.9284
Book-Crossing	0.7006	0.7105	0.7251	0.7408

#### 5.5. Cost of the Model

To discuss the cost of the model in training, we compare the propagation models RippleNet and Ripp-MKR, which are most relevant to the model in terms of time consumed. As shown in Table 5, our model is higher in time cost than the RippleNet with a simple structure because it introduces an attention mechanism as well as a propagation exploration framework. As for Ripp-MKR, it has more consumption because of the need for simultaneous multi-task learning during propagation. Since the graph neural network aggregation-based approach is very time consuming for graph training, it takes a long time to use KGCN.

**Table 5.** The cost of the model.

Time	RippleNet	EKPNet	Ripp-MKR	KGCN
MovieLens-1M	352.8 s	386.3 s + 252.1 s	924.1 s	1016.2 s
Book-Crossing	136.3 s	152.2 s + 122.4 s	524.2 s	734.1 s

## 6. Conclusions

In this article, we use the propagation of a knowledge graph to assist in the recommendation and design of a brand new EKPNet architecture. Compared with the existing propagation model, there are two main advantages. First, a new and effective asymmetric semantic attention mechanism is proposed to sample the knowledge graph. It improves the attention effect by distinguishing the preference semantics of the head and tail entities in the attention mechanism. Then, a communication feature exploration architecture is used to retain the communication features from shallow to deep. Features with different propagation depths enhance the memory and generalization capabilities during propagation, and finally use deep neural networks to explore feature intersections. Two structures are used to enhance the exploration of entity features in propagation. We conducted extensive experiments on two datasets in the real world and demonstrated the superiority of the EKPNet model through comparison with the baseline.

For future work, we will verify the effect of our model in more application scenarios. We mainly use offline recommendations in our experiments and the knowledge graph is stationary. To enhance the value of industrial applications, it is a promising research direction to study the propagation of recommender systems in dynamic knowledge graphs in the future. It can further obtain efficient online recommendation methods. In addition, our approach provides a new way of thinking for research in other heterogeneous networks such as social networks.

**Author Contributions:** H.Z.: Conceptualization, Methodology, Software, and Writing—original draft. Y.W.: Supervision and Project administration. C.C. and R.L.: Investigation, Writing—review and editing. S.Z.: Data curation and Resources. T.G.: Supervision and Project administration. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by National Key R&D Program (Demonstration of R&D and Application of Integrated Science and Technology Service Platform for Central Plains Urban Agglomeration), grant number 2018YFB1404500.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** We used open-source datasets and reference them in this article.

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

## References

1. Resnick, P.; Varian, H.R. Recommender systems. *Commun. ACM* **1997**, *40*, 56–58. [\[CrossRef\]](#)
2. Zhang, Y.; Abbas, H.; Sun, Y. Smart e-commerce integration with recommender systems. *Electron. Mark.* **2019**, *29*, 219–220. [\[CrossRef\]](#)
3. Zhang, H.; Ji, Y.; Li, J.; Ye, Y. A triple wing harmonium model for movie recommendation. *IEEE Trans. Ind. Inform.* **2015**, *12*, 231–239. [\[CrossRef\]](#)
4. Hu, Y.; Xiong, F.; Lu, D.; Wang, X.; Xiong, X.; Chen, H. Movie collaborative filtering with multiplex implicit feedbacks. *Neurocomputing* **2020**, *398*, 485–494. [\[CrossRef\]](#)
5. Sarwar, B.; Karypis, G.; Konstan, J.; Riedl, J. Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th International Conference on World Wide Web, Hong Kong, 1–5 May 2001; pp. 285–295.
6. Deng, S.; Huang, L.; Xu, G.; Wu, X.; Wu, Z. On deep learning for trust-aware recommendations in social networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2016**, *28*, 1164–1177. [\[CrossRef\]](#) [\[PubMed\]](#)
7. Aslanian, E.; Radmanesh, M.; Jalili, M. Hybrid recommender systems based on content feature relationship. *IEEE Trans. Ind. Inform.* **2016**, *99*, 1. [\[CrossRef\]](#)
8. Xu, Y.; Yang, Y.; Han, J.; Wang, E.; Zhuang, F.; Xiong, H. Exploiting the sentimental bias between ratings and reviews for enhancing recommendation. In Proceedings of the 2018 IEEE International Conference on Data Mining (ICDM), Singapore, 17–20 November 2018.
9. Guo, Q.; Zhuang, F.; Qin, C.; Zhu, H.; Xie, X.; Xiong, H.; He, Q. A survey on knowledge graph-based recommender systems. *IEEE Trans. Knowl. Data Eng.* **2020**, *50*, 937. [\[CrossRef\]](#)
10. Lehmann, J.; Isele, R.; Jakob, M.; Jentzsch, A.; Kontokostas, D.; Mendes, P.N.; Hellmann, S.; Morsey, M.; Van Kleef, P.; Auer, S.; et al. Dbpedia—a large-scale, multilingual knowledge base extracted from wikipedia. *Semant. Web* **2015**, *6*, 167–195. [\[CrossRef\]](#)
11. Suchanek, F.M.; Kasneci, G.; Weikum, G. Yago: A core of semantic knowledge. In Proceedings of the 16th International Conference on World Wide Web, Banff, AB, Canada, 8–12 May 2007; pp. 697–706.
12. Xu, W.; Gao, X.; Sheng, Y.; Chen, G. Recommendation System with Reasoning Path Based on DQN and Knowledge Graph. In Proceedings of the 2021 15th International Conference on Ubiquitous Information Management and Communication (IMCOM), Seoul, Korea, 4–6 January 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–8.
13. 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.
14. Wang, H.; Zhao, M.; Xie, X.; Li, W.; Guo, M. Knowledge graph convolutional networks for recommender systems. In Proceedings of the 2019 World Wide Web Conference, San Francisco, CA, USA, 13–17 May 2019; pp. 3307–3313.
15. 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, Turin, Italy, 22–26 October 2018; pp. 417–426.
16. Tang, X.; Wang, T.; Yang, H.; Song, H. AKUPM: Attention-enhanced knowledge-aware user preference model for recommendation. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 1891–1899.



17. Wang, Y.; Dong, L.; Li, Y.; Zhang, H. Multitask feature learning approach for knowledge graph enhanced recommendations with RippleNet. *PLoS ONE* **2021**, *16*, e0251162. [CrossRef] [PubMed]
18. Mahdy, A.M. Numerical solutions for solving model time-fractional Fokker–Planck equation. *Numer. Methods Partial. Differ. Equ.* **2021**, *37*, 1120–1135. [CrossRef]
19. Sun, Y.; Han, J.; Yan, X.; Yu, P.S.; Wu, T. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *Proc. VLDB Endow.* **2011**, *4*, 992–1003. [CrossRef]
20. Yu, X.; Ren, X.; Gu, Q.; Sun, Y.; Han, J. Collaborative filtering with entity similarity regularization in heterogeneous information networks. *IJCAI HINA* **2013**, *27*. Available online: [http://hanj.cs.illinois.edu/pdf/hina13\\_xyu.pdf](http://hanj.cs.illinois.edu/pdf/hina13_xyu.pdf) (accessed on 5 January 2022).
21. 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.
22. 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]
23. 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, 13–17 August 2016; pp. 353–362.
24. 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.
25. Ye, Y.; Wang, X.; Yao, J.; Jia, K.; Zhou, J.; Xiao, Y.; Yang, H. Bayes EMbedding (BEM) Refining Representation by Integrating Knowledge Graphs and Behavior-specific Networks. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, Beijing, China, 3–7 November 2019; pp. 679–688.
26. Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; Yakhnenko, O. Translating embeddings for modeling multi-relational data. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 5–10 December 2013; p. 26.
27. Cheng, Z.; Ding, Y.; He, X.; Zhu, L.; Song, X.; Kankanhalli, M.S. A3NCF: An Adaptive Aspect Attention Model for Rating Prediction. In Proceedings of the IJCAI, Stockholm, Sweden, 13–19 July 2018; pp. 3748–3754.
28. Mnih, V.; Heess, N.; Graves, A. Recurrent models of visual attention. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; pp. 2204–2212.
29. Han, K.J.; Prieto, R.; Ma, T. State-of-the-art speech recognition using multi-stream self-attention with dilated 1d convolutions. In Proceedings of the 2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), Singapore, 14–18 December 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 54–61.
30. 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.
31. Xiao, J.; Ye, H.; He, X.; Zhang, H.; Wu, F.; Chua, T.S. Attentional factorization machines: Learning the weight of feature interactions via attention networks. *arXiv* **2017**, arXiv:1708.04617.
32. Zhou, G.; Zhu, X.; Song, C.; Fan, Y.; Zhu, H.; Ma, X.; Yan, Y.; Jin, J.; Li, H.; Gai, K. Deep interest network for click-through rate prediction. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 1059–1068.
33. Yun, S.; Kim, R.; Ko, M.; Kang, J. Sain: Self-attentive integration network for recommendation. In Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, Paris, France, 21–25 July 2019; pp. 1205–1208.
34. Tu, K.; Cui, P.; Wang, D.; Zhang, Z.; Zhou, J.; Qi, Y.; Zhu, W. Conditional Graph Attention Networks for Distilling and Refining Knowledge Graphs in Recommendation. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management, Queensland, Australia, 1–5 November 2021; pp. 1834–1843.
35. Hu, Y.; Koren, Y.; Volinsky, C. Collaborative filtering for implicit feedback datasets. In Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, Pisa, Italy, 15–19 December 2008; pp. 263–272.
36. 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.
37. 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.
38. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
39. Harper, F.M.; Konstan, J.A. The movielens datasets: History and context. *Acm Trans. Interact. Intell. Syst.* **2015**, *5*, 1–19. [CrossRef]
40. Ziegler, C.N.; McNee, S.M.; Konstan, J.A.; Lausen, G. Improving recommendation lists through topic diversification. In Proceedings of the 14th international conference on World Wide Web, Chiba, Japan, 10–14 May 2005; pp. 22–32.