information

Article

A Multi-Granular Aggregation-Enhanced Knowledge Graph Representation for Recommendation

Xi Liu [1], Rui Song [2], Yuhang Wang [3] and Hao Xu [3,4,*]

[1] College of Software, Jilin University, Changchun 130012, China; liuxi19@mails.jlu.edu.cn
[2] School of Artificial Intelligence, Jilin University, Changchun 130012, China; songrui20@mails.jlu.edu.cn
[3] College of Computer Science and Technology, Jilin University, Changchun 130012, China; yuhangw19@mails.jlu.edu.cn
[4] Chongqing Research Institute, Jilin University, Chongqing 401123, China
* Correspondence: xuhao@jlu.edu.cn

Abstract: Knowledge graph (KG) helps to improve the accuracy, diversity, and interpretability of a recommender systems. KG has been applied in recommendation systems, exploiting graph neural networks (GNNs), but most existing recommendation models based on GNNs ignore the influence of node types and the loss of information during aggregation. In this paper, we propose a new model, named *A Multi-Granular Aggregation-Enhanced Knowledge Graph Representation for Recommendation* (MAKR), that relieves the sparsity of the network and overcomes the limitation of information loss of the traditional GNN recommendation model. Specifically, we propose a new graph, named the *Improved Collaborative Knowledge Graph* (ICKG), that integrates user–item interaction and a knowledge graph into a huge heterogeneous network, divides the nodes in the heterogeneous network into three categories—users, items, and entities, and connects the edges according to the similarity between the users and items so as to enhance the high-order connectivity of the graph. In addition, we used attention mechanisms, the factorization machine (FM), and transformer (Trm) algorithms to aggregate messages from multi-granularity and different types to improve the representation ability of the model. The empirical results of three public benchmarks showed that MAKR outperformed state-of-the-art methods such as Neural FM, RippleNet, and KGAT.

Keywords: knowledge graph; graph neural network; recommender system

Citation: Liu, X.; Song, R.; Wang, Y.; Xu, H. A Multi-Granular Aggregation-Enhanced Knowledge Graph Representation for Recommendation. *Information* **2022**, *13*, 229. https://doi.org/10.3390/info13050229

Academic Editor: Tudor Groza

Received: 18 March 2022
Accepted: 27 April 2022
Published: 29 April 2022

1. Introduction

With the rapid development of internet technology and the exponential growth of information resources on the network, users are facing the problem of information overload, and it is difficult to quickly retrieve the content in which they are interested [1]. As an effective method to alleviate the problem of information overload, recommender systems learn users' interest preferences and recommend personalized content by analyzing users' features, historical behavior data, and item features to help users retrieve the content they may be interested in [2].

Traditional recommendation methods are divided into three types: content-based recommendation, collaborative filtering (CF) recommendation, and hybrid recommendation [3]. Content-based recommendation mines other items with similar content as recommendations according to the items that users have selected or rated [4]. The collaborative filtering algorithm mines similar users or items by calculating the similarity between users or items and recommends items of interest to users [5]. The main idea of the hybrid recommendation algorithm is to combine the above recommendation methods to make full use of the information of users and items [6]. At present, the recommender system based on CF has been widely used, because it can effectively capture users' preferences and is

easy to implement. However, due to the exponential growth of network resources and users, CF-based recommendation has encountered challenges such as data sparsity and cold start [7].

To solve these problems, many studies have introduced side information, such as social networks [8], item attributes [9], and image information [10], into the recommender systems. Knowledge graph, an effective and commonly used type of side information, is able to alleviate data sparsity and cold starts, which are difficult and vital problems in recommender systems. Knowledge graph is a heterogeneous semantic network that is composed of nodes and edges. The nodes represent entities, and the edges represent various relationships between the entities. Knowledge graph has three obvious advantages: accuracy, diversity, and interpretability [11]. They introduce more semantic relationships that can accurately find deep-seated user interests, and they integrate a variety of different relationship types to ensure the diversity of recommendation results. Knowledge graph is an explicitly structured network. Information is transmitted according to the structure of the network, and all recommendation results are traceable. Therefore, knowledge graph can effectively alleviate the sparsity and cold start problems faced by traditional collaborative filtering systems in recommender systems and improve the recommendation effect to a certain extent.

The existing recommender systems based on knowledge apply knowledge graphs in three ways: embedding-based methods, path-based methods, and unified methods [12]. Based on the embedding method, firstly, the knowledge graph embedding method is used for pre-training in the KG to obtain the entity embedding representation in the KG, and then the entity embedding representation is used as the item embedding representation in the recommendation. CKE [10] learns the item embedding representation through TransR [13] with the participation of the KG. DKN [14] regards entity embedding and text embedding as different channels and generates new embedding representation through TransD [15]. However, the knowledge graph embedding algorithm is usually more suitable for graph applications, such as link prediction, rather than recommendation [16]; thus, the learning entity embedding representation is not intuitive and effective in representing the relationship between items. The path-based approach explores various connection modes between items in the KG to provide additional guidance for recommendation. A meta-graph-based recommendation KPRN [17] regards KGs as heterogeneous information networks and extracts the potential features based on meta-paths to represent the connectivity between users and items, along with different types of relationship paths. Path-based methods use KGs more naturally and intuitively, but they rely heavily on manually designed meta-paths. Based on the propagation method, the propagation is performed iteratively on the whole KG to find the recommended side information. RippleNet [18] has made great progress in KG-based recommendations. It regards the user's historical interest as the seed set in the KG and then iteratively expands the user's interest along the KG link to find its layered potential interest in candidate items. KGCNs [19] use graph convolutional networks (GCNs) [20] to obtain item-embedding representation through their neighbors in the KG, which shows that neighbor information can effectively improve the accuracy of recommendation tasks. However, both methods ignore the contribution of collaborative information to the user and item representation, resulting in insufficient embedded representations of the user and item.

To sum up, combining knowledge graphs with GNN-based models for recommendation presents two challenges: (1) The existing recommendation algorithms based on knowledge graphs often ignore the contribution of high-order collaborative information to the representations of users and items, resulting in an insufficient embedded representation of the users and items. Therefore, we investigated how to consider both user commodity interaction behavior and the KG, enhance the high-order connectivity of the graph, and better model the embedded representation of users and items to improve the recommendation performance. (2) The previous traditional aggregation methods, such as mean pooling, maximum pooling, and matrix multiplication, may cause heterogeneous

messages to offset each other and introduce information loss. Therefore, we investigated how to construct a new aggregation method to improve the expression ability of the model.

To address the above problems, we propose a multi-granular information aggregation recommendation (MAKR) model to enhance the collaborative information and knowledge graph information. Figure 1 shows the model framework. Firstly, the model pre-trains the node embedding in the graph based on the DeepWalk model to obtain the preliminary node embedding [21]. The model introduces first-order collaborative information and high-order collaborative information and uses attention mechanisms [22], FM [23], and transformer [24] to extract important information from different dimensions to obtain enhanced collaborative information, which is used to fully learn the influence between the categories and all nodes. Then, the edges are connected according to the similarities of the user–user and item–item, and the graph is enhanced. This method can effectively reduce the sparsity of the graph and enhance the representation of the user portrait and commodity features. Finally, the enhanced collaboration information and knowledge graph information are combined through the aggregator to obtain the final representation of users and items. The main contributions of this paper are summarized as follows:

We propose a novel information aggregator on the GNN model that jointly considers type-aware attention, fine granular transformer, and coarse granular FM aggregators, greatly improving the representation ability of the model.

1. The graph is enhanced to relieve sparsity. The edge is connected according to the similarity threshold between the user–user and item–item in the graph, which combines user–item interaction and item attributes. The strategy can better model user portraits and item features;
2. We propose a new model, MAKR, based on the GNN in a knowledge graph for recommendation tasks. Furthermore, we conducted experiments on three top-*N* recommender data sets with different settings that indicate that MAKR obtained a state-of-the-art position in the top-*N* recommendation.

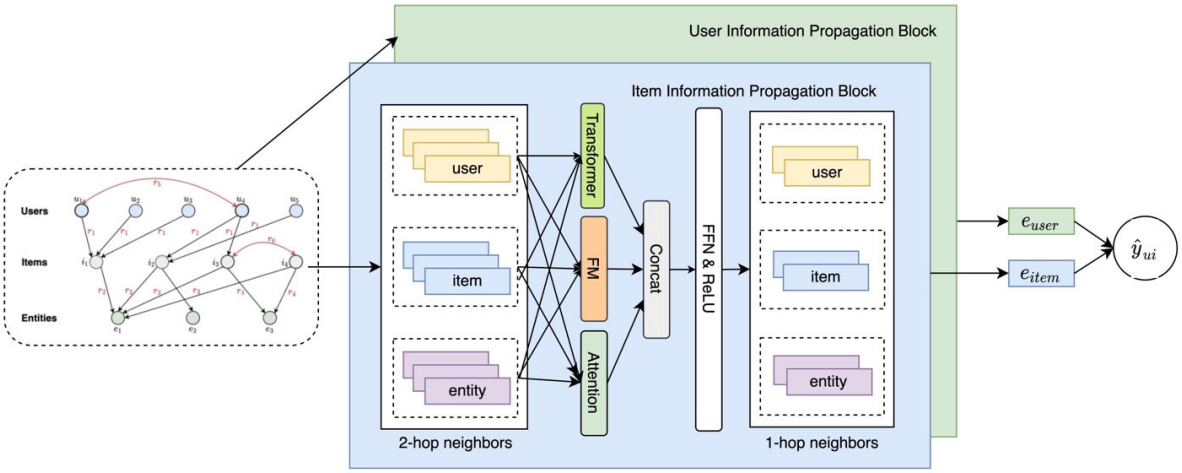

Figure 1. The whole framework of the proposed MAKR model.

The rest of the paper is organized as follows. First, Section 2 introduces the related works from three aspects: knowledge graphs, graph neural networks, and GNN-based recommender systems. Section 3 introduces our method in detail, which includes graph construction, the three main parts of our models, and optimization. Next, we describe the three data sets, baseline models, evaluation metrics, and experimental settings in detail in Section 4. Section 5 shows the overall test performance of our models and baseline models and reports the experimental results in detail. Finally, we summarize this research and discuss the prospects for future research in Section 6.

2. Related Work

2.1. Graph Neural Network

Traditional neural networks, such as CNN and RNN, cannot directly process the graph structure data. The nodes in the graph must be transformed into the sequence form acceptable to CNN and RNN. However, this will lose the structure's information in the graph and the computing resources cannot be borne.

A graph neural network is a deep neural network based on graph structure data that can fully learn the characteristics and patterns of graph structure data so as to complete the next tasks: node classification, link prediction, inner product scoring, and recommendation. DeepWalk [21] uses the random walk method to learn the representation of nodes in the graph structure for the first time. A graph revolution network (GCN) [20] uses convolution to carry out classification tasks, which can be divided into two types: spectral based and spatial based. GraphSAGE [25] optimizes GCN by sampling neighbors with nodes as the center so that it can be trained in large-scale graph data. Moreover, GraphSAGE can perform inductive learning and learn nodes that have not been seen before. Graph attention network（GAT) [26] adds an attention mechanism to GraphSAGE. The importance of neighbors is measured in the process of neighbor aggregation.

2.2. Knowledge Graph

Knowledge graph originates from the semantic web, which is a special heterogeneous map that contains rich semantic information and structural information. Knowledge graph is represented in the form of triples, including head nodes, tail nodes, and edges connecting the two nodes. The head node and tail node represent the entities in the objective world, and the edge represents the relationship between the two entities. Since Google put forward the concept of the knowledge graph in 2012, researchers have explored knowledge graphs from various angles. In particular, major companies have built many large knowledge graphs at an early stage, such as Freebase [27], DBpedia [28], Yago [29], and Nell [30], and have made many successful applications. Nowadays, knowledge graph plays an important role in question-and-answer systems, intelligent customer service, information retrieval, recommender systems, and other fields.

Different from the ordinary graph structure, the edges of knowledge graphs have practical meaning, and the types of edges between each pair of nodes may be different. N. Kipf pioneered the use of GCN for multi-relational graphs, such as knowledge graphs, while all previous graph neural network models can only model graphs with a single relationship [31]. In order to deal with different relationships in the knowledge graph, the relationship-aware subgraph aggregator divides the graph into multiple subgraphs, and the subgraph contains only one relationship type. Each subgraph has an aggregator, which extracts the information of the subgraph to produce a context representation containing relational information. KGCN [19] and KGAT [32] optimize the neighborhood graph into a weighted graph in the knowledge graph. These weights capture rich semantic information about the relationship between the different nodes. At the same time, the interpretability of the depth recommender system is improved by weighting the contributions of different nodes near the target node.

2.3. Traditional Recommender System

Recommendation system can effectively solve the problem of information overload. Generally speaking, recommendation methods are divided into three kinds: content-based recommendation, collaborative filtering (CF) recommendation and hybrid recommendation.

Content-based recommendation system attempts to recommend items similar to those a given user has liked in the past [33]. Nguyen considered that background data are significant to recommender systems; they employed DBpedia and Freebase to evaluate the fitness for content-based recommendation tasks in the music domain [34]. The premise

for content-based recommendation system to give a good recommendation is that the information in the content is enough to distinguish users' interest in the items. The content-based recommendation system will not work if the information in the content is insufficient [35].

CF assumes that users may be interested in items selected by people who share similar interaction records. To implement CF-based recommendation, interaction data from multiple users and items are required, which further forms the user-item interaction matrix [12]. Collaborative filtering methods are divided into user-based methods and item-based methods. In fact, similarity calculation is the common point of user-based methods and item-based methods, but the similarity calculation method is different. The former is inferred from the preference of user history, and the latter is based on the attribute feature information of the item itself.

The traditional method is difficult to learn the high-order structure information in the data, while the graph neural network GNN uses the message passing mechanism to integrate the neighbor information, so that the node can obtain the high-order neighbor information. Therefore, GNN has been widely used in recommendation systems in recent years, and it has become the most advanced method. MHCN [36] uses GCN to spread information on the constructed hypergraph to obtain high-order relations. DiffNet [37] uses different weights on GNN to measure the importance of different friends in social networking, which has achieved good results.

2.4. GNN-Based Recommender System

With the rapid development of GNN and its excellent performance in node classification, many researchers try to apply GNN to recommender systems.

The data in a recommender system is generally divided into three types: user–item interaction, item sequence, and other side information such as the social relationship. Knowledge graph user–item interactions can be regarded as a bipartite graph, the item sequence can be regarded as a sequence graph, and the side information can naturally be regarded as graph structure data. Therefore, we can see that most of the data in a recommender system (in essence, it can be seen as graph structure data) are suitable for GNN. In addition to data, for the specific learning process, GNN can explicitly encode cooperative signals through node aggregation to enhance the representation and learning ability of users and items. Compared to other models, GNN's modeling of multi-hop information is more flexible and convenient [38]. In short, the data structure of a recommender system is very suitable for transformation into a graph structure, and GNN has a strong learning ability in the representation and learning of graph data.

NGCF [39] and LightGCN [40] enhance user representation by using user–item interactive records and enhance item representation by using interactive user records. SR-GNN [41] is the migration of items through iterative propagation based on sequence diagrams to learn representative item representation. KGCN [19] and KGAT [32] integrate a knowledge graph into recommender systems, which leads to two benefits: (1) rich semantic associations between items can improve the effect of item representation; (2) the interpretability of the recommended results is enhanced.

In MAKR, we not only use interactive data but also knowledge graph information. Moreover, we implemented a novel GNN-based recommender system that combines factorization machine, attention mechanism, and transformer layers for node aggregation with different feature fields.

3. Methods

In this section, the detailed method is introduced. The MAKR model is mainly divided into five modules: (1) Graph construction—builds user–item interaction; the item attributes information into the user–item bipartite graph and knowledge graph, integrating and improving the first graph. (2) Embedding layer—the initial feature vectors of users, items, and entities are obtained through random initialization or pre-training

embedding. (3) Information propagation layer—carries out message propagation for nodes in the self-defined graph neural network module and updates the node representation. (4) Prediction layer—outputs the prediction results of the model according to the final user and item node representation. (5) Optimization: We used BPR loss and L2 regularization terms as loss functions to optimize our model. The message propagation layer is the focus of the model in this chapter. The message propagation layer focuses on the interactive information between neighbor nodes and carries out interactive aggregation with different granularity so as to reduce the information loss during aggregation as much as possible.

Next, the structure of each layer of the MAKR model in this paper is introduced in detail, and each layer is described in Sections 3.2–3.4.

3.1. Graph Construction

- **User–Item Bipartite Graph:** User historical behavior, purchases, and clicks are important and widely used in recommender systems. We constructed the user–item interaction behavior as a user–item bipartite graph, G_1, with the user set on the left and the item set on the right. Users and items are connected to each other. There is no connection between users and no connection between items. G_1 is defined as $\{(u, y_{ui}, i)|u \in \mathcal{U}, i \in \mathcal{I}\}$, where $\mathcal{U}$ is the user set and $\mathcal{I}$ is the item set, and $y_{ui} = 1$ if there is a user–item interaction; otherwise, $y_{ui} = 0$.
- **Knowledge Graph:** User–item interaction information is sparse. In order to enrich the data, many studies have tried to add item attributes or external knowledge into the recommender system as side information. Here, we formed an item attribute knowledge graph, G_2, by integrating the item, its attributes, and the relationship between them. This knowledge graph is composed of triples, expressed as $\{(h, r, t)|h, t \in \mathcal{E}, r \in \mathcal{R}\}$, where h represents the head entity, t represents the tail entity, and r represents the relationship between them. For example, $(Timothy\ Donald\ Cook, Manage, Appl)$ states the fact that Timothy Donald Cook manages Apple. Note that $\mathcal{R}$ contains relations in both the canonical direction (e.g., $Manage$) and inverse direction (e.g., $ManagedBy$). Moreover, establishing a set of item–entity alignments is necessary. Alignments are depicted as $\mathcal{A} = \{(i, e)|i \in \mathcal{I}, e \in \mathcal{E}\}$, where (i, e) indicates that item i aligns with e in the knowledge graph.
- **Collaborative Knowledge Graph (CKG):** In order to enrich the data of the recommender system and enhance the expression ability of the model, we integrated the user–item bipartite graph and item attributes knowledge graph into one graph to build a more complete graph. Here, we define the concept of the CKG, which encodes user behaviors and item knowledge as a unified relational graph [32]. Firstly, we represent the interaction behavior of each user–item pair as a triple $\{u, interact, i\}$, where u represents the user, i represents the item, and $interact$ is the relationship between the user and item. Then, based on the item entity alignment mentioned above, G_1 and G_2 can seamlessly form a unified CKG = $\{(h, r, t)|h, t \in \mathcal{E}', r \in \mathcal{R}'\}$, where $\mathcal{E}' = \mathcal{E} \cup \mathcal{U}$ and $\mathcal{R}' = \mathcal{R} \cup \{Interact\}$.
- **Improved Collaborative Knowledge Graph (ICKG):** Although CKE has strong presentation ability, including both user interaction information and item knowledge, in the bipartite graph of user interaction, there is neither an edge between the users nor between goods, ignoring the influence of users and goods. Whether the user–user is connected or not is calculated by the similarity of the two users:

$$sim(i, j) = \alpha_1 \frac{\langle Y_i, Y_j \rangle}{\|Y_i\| \cdot \|Y_j\|} + \alpha_2 \frac{\langle D_i, D_j \rangle}{\|D_i\| \cdot \|D_j\|} \quad (1)$$

where Y_i represents the user–item interaction vector of the user i (click 1, otherwise 0), and D_i represents the DeepWalk pre-training vector of the user i (described in detail in Section 3.2). $\alpha_1 = \alpha_2 = 0.5$, that is, the similarity between two users, is determined by the

user's historical behavior and the DeepWalk vector. After calculating the similarity, it takes the K neighbors with the greatest similarity to each user.

Whether the item–item is connected or not is the same for the user–user. Finally, according to the value of K, we connect the user–user and item–item on the CKG to form an ICKG. As shown in Figure 2, the red arrow r_5 indicates the connected edges of similar users, and r_6 indicates the connected edges of similar items.

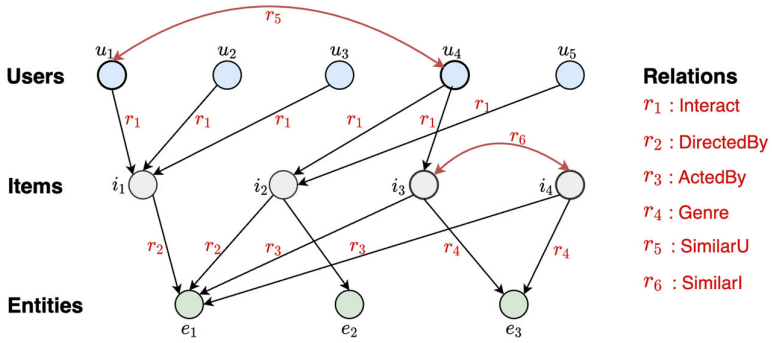

Figure 2. Improved collaborative knowledge graph (ICKG). The red arrow r_5 indicates the connected edges of similar users, and r_6 indicates the connected edges of similar items.

3.2. Embedding Layer

Graph-embedding representation uses the method of deep learning to represent the adjacency relationship in the topological graph, with vectors embedded in the low-dimensional space [25]. DeepWalk is a typical data mining method in graph structure [21].

Here, we employed DeepWalk, widely used in graphs, on a user–item bipartite graph. To be more specific, in a user–item bipartite graph, each node is randomly walked five times to obtain five random walk sequences, and then trained with a word2vec algorithm with skip-gram and hierarchical SoftMax [42,43]. Thus, the vector of each user and item can be obtained as the projection-embedding matrix.

All heterogeneous nodes need to be projected into the feature space. For node u and its neighbor $\mathcal{N}_k$, the graph construction will add self-connection; thus, the node itself is also its own neighbor. Then, each node can be represented as $\{\bar{u}_k, \bar{\imath}_k, \bar{e}_k\}$ according to its node type, which refers to the one-hot representation of the user and item, respectively. After mapping the one-hot representation through the projection embedding matrix, the embedded feature representation of nodes is obtained.

$$h_u = P_u \, \bar{u} \tag{2}$$

Here, h_u represents the user feature embedding vector, $\bar{u}$ represents the user one-hot embedding, and $P_u \in \mathcal{R}^{d_u \times n_u}$ denotes the projection-embedding matrix. n_u is the number of user nodes, and d_u is the dimension of embedding. The projection-embedding matrix uses the matrix obtained by the above DeepWalk algorithm. Then, we obtain users' feature embedding. We use the same method on the item.

3.3. Embedding Propagation Layers

The message propagation layer is the focus of the MAKR model, and the overall structure is shown in Figure 3. In addition to learning the attention scores among the user, item, and entity nodes through the attention mechanism, this layer also uses FM and transformer to learn the cross-features of the different granularity between neighbor nodes. FM focuses on the second-order cross-feature, and transformer focuses on the multi-order cross-feature.

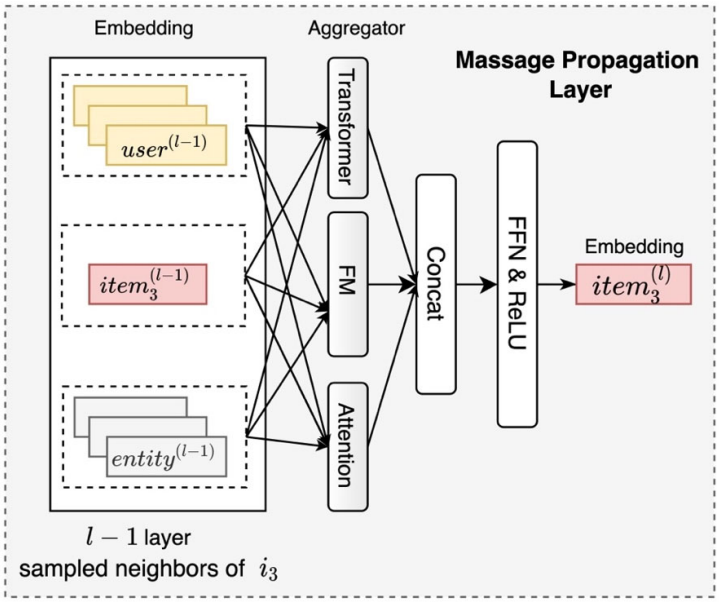

Figure 3. Message propagation layer.

3.3.1. Aggregator Based on Attention Mechanism

After sampling the neighbors of the user or item node, the next step is to aggregate these neighbor features to obtain the node's new feature representation based on the neighbor information. For example, to aggregate node u's neighbor features, we use the aggregator based on an attention weighting mechanism:

$$e_{\mathcal{N}_u}^{Att} = \sum_{j \in \mathcal{N}_u} \alpha_{u \leftarrow j} e_j \quad (3)$$

where $\alpha_{u \leftarrow j}$ is the weight assigned by node u to neighbor j, which indicates the importance of neighbor j to u. The neighbor nodes include the user, item, and entity. One of the most common and simple ways to calculate this weight is to set $\alpha_{u \leftarrow j} = \frac{1}{|\mathcal{N}_u|}$ as shown in Equation (3). But this method assumes that all neighbors have the same importance, which is obviously unreasonable.

In order to solve the above problems, we use the attention mechanism to adaptively calculate the attention weight. Informative neighbor nodes will help to provide more significant features, while irrelevant neighbors will be ignored to a large extent. When calculating the attention weight at the node level, the model takes into account the information of the node representation and relation type as shown in Equation (4). The attention network not only knows the central node and neighbor nodes but also considers the relation types between them, which makes the whole attention modeling process more comprehensive.

$$\alpha'_{u \leftarrow j} = \mathrm{v}^{\mathrm{T}} \mathrm{LeakyReLU}(\mathrm{W}_1[h_u \parallel h_{rel}] + \mathrm{W}_2 h_j + \mathrm{b}) \quad (4)$$

Here, $\parallel$ represents the concatenate operation, h_u and h_{rel} are the embedding feature of the central node and neighbor j, and $e_{relation}$ is the embedding feature of the ID of the relation $u \leftarrow j$. In the graph construction process, the relation type ID is processed and used as the attribute of the relation. In addition, $\mathrm{W}_1, \mathrm{W}_2$, b, and v^{T} are the trainable parameters of the model. The model uses the relation-aware attention aggregator, which can learn the hidden layer features of nodes corresponding to different types of edges more finely. Finally, the attention weight, $\alpha_{u \leftarrow j}$, needs to be normalized using SoftMax to obtain $\alpha'_{u \leftarrow j}$:

$$\alpha_{u\leftarrow j} = \text{softmax}_j\left(\alpha'_{u\leftarrow j}\right) = \frac{\exp\left(\alpha'_{u\leftarrow j}\right)}{\sum_{k\in\mathcal{N}_u}\exp\left(\alpha'_{u\leftarrow k}\right)} \quad (5)$$

Generally, existing heterogeneous graph convolution models usually use the function based on pooling to update node information, such as average pooling, maximum pooling, and attention pooling, which will lose the node information and ignore the important cross-features in a recommender system. Therefore, we propose an information updating method based on FM and transformer to capture multi-granularity cross-features.

3.3.2. Aggregator Based on Factorization Machine

FM is a factorization machine proposed in reference [23], which is mainly used to learn the interactive features of the recommendation model. In addition to the linear first-order interaction features between features, the FM model also models paired second-order interaction features as the inner product of their potential feature vectors.

FM can capture second-order cross-features more effectively than previous methods, especially under conditions of sparse data sets. In the previous methods, only when the features x_i and x_j appear in the same record can the parameters of the cross action be trained. In FM, the cross-feature is learned through the inner product of the hidden vector of its feature. With this flexible design, FM can train the hidden vectors, V_i or V_j, when the features x_i or x_j appear in the data record. Therefore, FM can better learn the cross-features that never appear or rarely appear in the training data. In this model, the output of the neighbor nodes through the FM aggregator is the sum of several second-order inner product elements:

$$y_{\mathrm{FM}} = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} < \mathrm{V_i}, \mathrm{V_j} > x_i x_j \quad (6)$$

where $\mathrm{V_i} \in \mathcal{R}^k$, and K is the dimension of the user-defined hidden vector, which is a hyperparameter. The second-order inner product element $< \mathrm{V_i}, \mathrm{V_j} >$ represents the influence of the feature interaction.

As shown in Figure 4, FM uses the neighbor information of u_1 to learn the cross-feature of node u_1. Suppose that the neighbors of user u_1 are $\{u_1, e_1, e_2, i_1, i_2\}$, and the corresponding features are expressed as $\{h_1, h_2, h_3, h_4, h_5\}$. We concatenated these neighbor features and then obtained a new representation of user u_1's second-order cross neighbor aggregation through the FM layer.

$$h_{\mathcal{N}_u}^{FM} = \sum_{i=1}^{4}\sum_{j=i+1}^{5} h_1 \odot h_2 \quad (7)$$

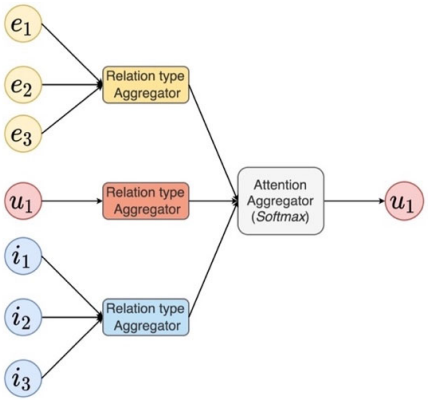

Figure 4. Attention aggregator.

There are still deficiencies. As shown in Figure 5, FM only performs a second-order cross on neighbor features, and the learned cross-features are coarse-grained. This model further uses transformer to learn more fine-grained features.

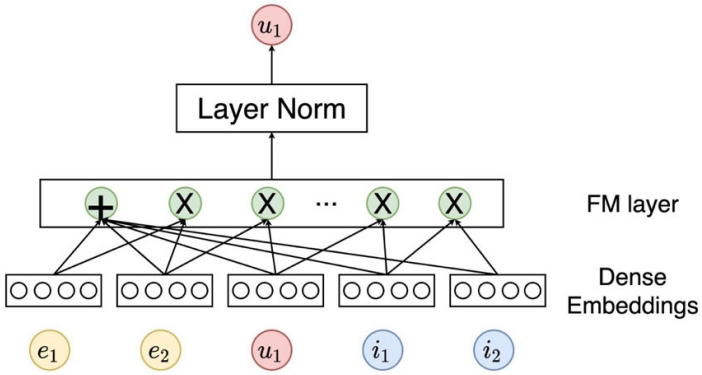

Figure 5. FM aggregator.

3.3.3. Aggregator Based on Transformer

The transformer model was first applied in machine translation tasks [24]. Later, with the emergence of Bert, a pre-training language model based on transformer, transformer was gradually applied to other fields. Transformer consists of two parts: the encoder and the decoder. This model mainly uses the encoder part to encode sequence information.

In this model, transformer is applied to the neighbor aggregation of nodes in an ICKG graph. As shown in Figure 6, the neighbors of the nodes do not have the concept of order, so the transformer's position embedding is not used. The core of the transformer is the multi-head self-attention layer [44], and then the residual connection [45] and layer normalization mechanism [46] are added. Next, we introduce the principle of transformer cross-aggregation by the example of item neighbor cross-aggregation.

Taking a user node u in the ICKG as an example, the features of the sampling neighbor nodes of the user node u, which include the user, item, and entity nodes, are expressed as $F_u = \{h_1, h_2, \ldots, h_n\}$. The query, key, and value of self-attention are all from F_u. Then, it is obtained through a linear transformation:

$$Q = W^Q F_u, \quad K = W^K F_u, \quad V = W^V F_u \tag{8}$$

where W^Q, W^K, and W^V are linear transformation matrices, which are the learnable parameters of the model, and n is the number of neighbors. Then, the self-attention score is calculated based on the query, key, and value:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{9}$$

where T is matrix transformation operation, and K^T represents the transpose matrix of K.

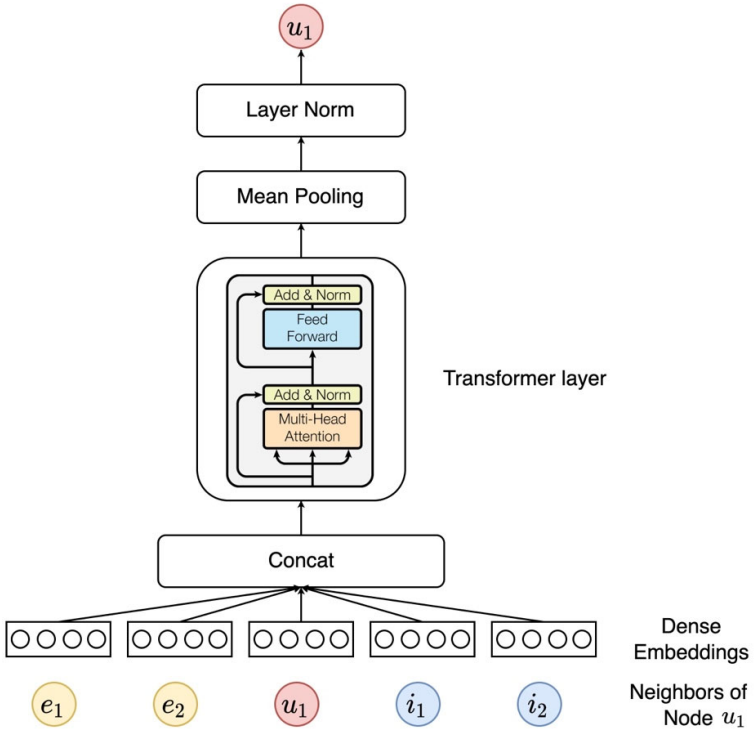

Figure 6. Transformer aggregator.

In order to further extract effective information from different potential subspaces, this model uses the multi-head self-attention mechanism for modeling:

$$H_u = concat(\, head_1, \ldots, head_h) W^O \tag{10}$$

$$head_i = Attention(W_i^Q Q, W_i^K K, W_i^V V) \tag{11}$$

Finally, the model obtains the node representation of the output of the transformer neighbor aggregator through the average pooling layer and then normalizes the layer to make the model training more stable and accelerate the convergence speed.

$$h_{\mathcal{N}_u}^{Trm} = Layer\_norm\,(Average\_pooling(H_u)) \tag{12}$$

In addition, the transformer aggregator stacks multiple transformer layers to learn more fine-grained neighbor cross-features.

After the attention aggregation, FM aggregation, and transformer aggregation of the node's neighbor information, the model concatenates the outputs of the three and obtains the final node representation through a linear transformation.

$$h_u^{(l+1)} = ReLU(Dense\left(concat\left(h_{\mathcal{N}_u}^{Att}, h_{\mathcal{N}_u}^{FM}, h_{\mathcal{N}_u}^{Trm}\right)\right)) \tag{13}$$

3.4. Model Prediction

After the customized attention aggregation and multi-granularity interactive aggregation information message propagation layer, the model can obtain the final feature representation of each node. The output of each layer of the user is $\{h_u^{(0)}, h_u^{(1)}, \cdots h_u^{(L)}\}$, and the output of each layer of the item is $\{h_i^{(0)}, h_i^{(1)}, \cdots h_i^{(L)}\}$.

The output of each layer is a message aggregation of a tree structure with a depth of L, with u or i as the root. Therefore, the output of different layers emphasizes the connectivity information of the different layers. Therefore, we refer to the layer aggregation

mechanism [47] to concatenate the output of each layer. In this way, we obtain the final u, i representation.

$$h_u^* = h_u^{(0)}||h_u^{(1)}||\cdots||h_u^{(L)}, \quad e_i^* = h_i^{(0)}||h_i^{(1)}||\cdots||h_i^{(L)} \tag{14}$$

where $||$ is a concatenation operation. Finally, we use the vectors of the users and items to carry out an inner product operation to obtain the matching scores of the end users and goods:

$$\hat{y}(u,i) = {h_u^*}^T h_i^* \tag{15}$$

3.5. Optimization

In order to make the ranking ability better, we adopted BPR loss [48] to optimize our proposed model, which is based on Bayesian personalized ranking. The idea of BPR loss is to maximize the difference between the scores of positive samples and negative samples. The specific formula is as follows:

$$\mathcal{L} = \sum_{(u,i,j)\in\mathcal{O}} -\ln\sigma\left(\hat{y}(u,i) - \hat{y}(u,j)\right) \tag{16}$$

where $\mathcal{O} = \{(u,i,j)|(u,i) \in \mathcal{R}^+, (u,j) \in \mathcal{R}^-\}$ indicates the training set. $\mathcal{R}^+$ indicates the observed (positive) interactions between user u and item j, while $\mathcal{R}^-$ is the sampled unobserved (negative) interaction set; $\sigma(\cdot)$ is the sigmoid function.

To alleviate over-fitting, we added the L2 regularization term to the loss function so as to make the model not too complicated and improve the accuracy of the model generalization prediction.

$$\mathcal{L}_{MAKR} = \mathcal{L} + \lambda\|\Theta\|_2^2 \tag{17}$$

where λ represents the hyperparameter of the L2 regularization term, and Θ represents the parameters that the model needs to be trained.

4. Materials and Experiments

In this section, we describe our data sets, baseline models, and experimental settings in detail.

4.1. Data sets

We conducted experiments on three data sets including Yelp2018, Amazon-Book, and Last-FM. A detailed description of each data set is listed in Table 1.

Table 1. Statistics of the data sets.

		Yelp2018	Last-FM	Amazon-Book
User–Item Interaction	Users	45,919	23,566	70,679
	Items	45,538	48,123	24,915
	Interactions	1,185,068	3,034,796	847,733
Knowledge Graph	Entities	90,961	58,266	88,572
	Relations	42	9	39
	Triplets	1,853,704	464,567	2,557,746

- **Yelp2018:** This data set is about hotel management. We considered restaurants and bars as the items. We collected the data set from the 2018 edition of the Yelp challenge (https://www.yelp.com/dataset/challenge) (accessed on 5 March 2022);
- **Amazon-Book:** The Amazon-Book data set records the book information of Amazon and users' ratings of Amazon books. Here, we viewed the books as the items. (http://jmcauley.ucsd.edu/data/amazon) (accessed on 5 March 2022);
- **Last-FM:** Last-FM is a data set about the sequence of users listening to songs that is provided by the Last-FM online music system. We took tracks as the items. We used

the subset of the data set from January 2015 to June 2015. (https://grouplens.org/datasets/hetrec-2011/) (accessed on 5 March 2022).

To ensure the quality of the data sets, we used users and items that had at least 10 interactions.

4.2. Baselines

To demonstrate the effectiveness, we compared our proposed model with the supervised learning model without a knowledge graph (i.e., FM and NFM), an embedding-based model (i.e., CKE), a path-based model (i.e., MCRec), and a unified model (i.e., RippleNet and KGAT) as follows:

- **FM** [23]: Factorization machine (FM) is a classical recommendation method that performs second-order interaction on all input features. Here, the IDs of a user, an item, and the knowledge consisted of the entities as input features;
- **NFM** [49]: NFM brings DNN into FM methods, learning more information through the nonlinear structure. Here, we used one hidden layer on the input features as suggested in [49];
- **CKE** [10]: This is an embedding-based method that uses an item's attributes graph as the knowledge graph. The latent vector is encoded with the TransR algorithm;
- **CFKG** [50]: CFKG considers user behaviors as a relation in the user–item KG, which includes the user–item interaction and item attributes;
- **RippleNet** [18]: RippleNet merges the embedding-based and path-based methods to enhance user representations by propagating the user's preferences from historical interests along the path in the KG;
- **GC-MC** [51]: GC-MC applies the GCN method to the user–item bipartite graph. The model consists of three parts: ordinal mixture GCN, dense, and bilinear mixture.
- **KGAT** [32]: This method is a state-of-the-art knowledge graph-based model that applies the attention mechanism to the KG convolution for modeling high-order relations.

4.3. Evaluation Metrics

The method outputs the user's preference scores over all the items except the positive ones in the training set. To evaluate the effectiveness of top-*N* recommendation and preference ranking, we adopted two widely used evaluation metrics [52,53]: recall@k and ndcg@k. By default, we set k = 20. We report the average metrics for all users in the test set.

$$recall = \frac{\sum_{u \in U} |R(u) \cap T(u)|}{\sum_{u \in U} |T(u)|} \tag{18}$$

Here, $R(u)$ represents the top-*N* recommendation list made to the user according to the user's behavior on the training set, and $T(u)$ represents the set of items actually selected by the user after the system recommends items to the user.

Normalized discounted cumulative gain (NDCG) was used to evaluate the sorting results. It is common in search and recommendation tasks. IDCG means the best-ranked DCG.

$$DCG_K = \sum_{i=1}^{k} \frac{rel(i)}{\log_2(i+1)} \tag{19}$$

$$NDCG = \frac{DCG}{IDCG} \tag{20}$$

4.4. Experiment Settings

For all the data sets, we removed the user–item interactions that were used less than 10 times for the quality of the data sets. For each user, we randomly selected 80% of the interacted items as the training set, 10% of the interacted items as the validation set, and

the remaining 10% as the testing set. We optimized the models with mini-batch Adam, where the batch size was fixed at 1024 and the learning rate was tuned among {0.05, 0.01, 0.005, 0.001}. The negative items were tuned among {1, 5, 10, 15, 20}, and the dropout ratio was tuned in {0.1, 0.3, 0.5, 0.7, 0.9}. Moreover, we adopted an early stopping strategy that interrupted the training process if recall@20 on the validation set did not increase for 10 successive epochs. We implemented our model in *Pytorch* and DGL, which is a GNN framework for effectively processing and training graph data.

5. Results and Discussion

In this section, we compare our model with seven baseline models and conduct an ablation study to analyze the influence of the different parts of the model.

5.1. Overall Comparison

Table 2 shows the top-*N* recommended performance of MAKR on three data sets compared with the baseline model, where n = 20. Obviously, MAKR achieved the best performance on all the data sets. MAKR was also higher than the strongest baseline KGAT; the ndcg@20 values of Amazon-Book, Last-FM, and Yelp2018 increased by 9.54%, 5.21%, and 5.19%, respectively. By superimposing multiple customized message propagation layers, MAKR can explore high-order connectivity in an abundant way so as to effectively capture collaborative signals. This verifies the importance of capturing collaborative information to transfer knowledge. In addition, compared to KGAT, MAKR proved the effectiveness of the multi-granularity cross-aggregation neighbor mechanism. On the basis of the attention aggregation neighbor, FM and transformer were further used to aggregate neighbor characteristics with different granularity.

Table 2. Overall performance comparison.

	Amazon-Book		**Last-FM**		**Yelp2018**	
	Recall	**Ndcg**	**Recall**	**Ndcg**	**Recall**	**Ndcg**
FM	0.1345	0.0886	0.0778	0.1181	0.0627	0.0768
NFM	0.1366	0.0913	0.0829	0.1214	0.0660	0.0810
CKE	0.1343	0.0885	0.0736	0.1184	0.0657	0.0805
CFKG	0.1142	0.0770	0.0723	0.1143	0.0522	0.0644
RippleNet	0.1336	0.0910	0.0791	0.1238	0.0664	0.0822
GC-MC	0.1316	0.0874	0.0818	0.1253	0.0659	0.0790
KGAT	0.1489	0.1006	0.0870	0.1325	0.0712	0.0867
MAKR	**0.1571**	**0.1102**	**0.0930**	**0.1394**	**0.0775**	**0.0912**
% Improvement	5.51%	9.54%	6.90%	5.21%	8.85%	5.19%

The performance of GC-MC in the Last-FM and Yelp2018 data sets was equivalent to that of RippleNet. While introducing high-order connectivity into user and item representation, GC-MC abandons the semantic relationship between the nodes. RippleNet can use relationships to explore users' interests and preferences. Compared to the FM model, RippleNet improved the effect and verified that merging multi-hop adjacent nodes can effectively enrich the user's representation. This also reflects that the high-order connectivity or neighbor modeling was effective.

The FM and NFM methods achieved a better performance than CFKG and CKE in most cases, which shows that regularization-based methods may not make full use of item knowledge. In order to enrich the representation of items, FM and NFM use the embedding of their associated entities, while CFKG and CKE only use the embedding of their aligned entities. In addition, the cross-function in FM and NFM is actually the second-

order connection between the users and entities, while CFKG and CKE model the connection on the granularity of triples to maintain the high-order connectivity.

5.2. Parameter Sensitivity Analysis

5.2.1. Effect of Depth

We changed the depth of the MAKR model to study the effect of using multiple message propagation layers. Specifically, the influence of model depth was explored in the range of {1, 2, 3, 4}. In this paper, MAKR-1 is used to represent a one-layer model, and more layers are represented by similar symbols. Table 3 shows the impact of the number of message propagation layers on the model effect. Summarizing the results in Table 3, we can see that increasing the depth of MAKR can effectively improve performance. Obviously, MAKR-2 and MAKR-3, in the three data sets of Amazon-Book, Last-FM, and Yelp2018, all achieved consistent improvement over MAKR-1. We attribute these improvements to the effective modeling of high-order connectivity among users, items, and entities, which are second-order and third-order connectivity, respectively. When stacking another layer on MAKR-3, it can be observed that MAKR-4 only achieved a slight improvement. This suggests that considering the third-order relationship between the user–item entity may be sufficient for learning the representation of the user and item.

Table 3. Effect of embedding propagation layer numbers (L).

	Amazon-Book		**Last-FM**		**Yelp2018**	
	Recall	**Ndcg**	**Recall**	**Ndcg**	**Recall**	**Ndcg**
MAKR-1	0.1450	0.1007	0.0843	0.1302	0.0705	0.0810
MAKR-2	0.1525	0.1053	0.0902	0.1324	0.0723	0.0873
MAKR-3	0.1571	0.1102	**0.0930**	**0.1394**	**0.0775**	0.0912
MAKR-4	**0.1575**	**0.1110**	0.0923	0.1387	0.0772	**0.0920**

5.2.2. Effect of Aggregators

Table 4 shows the impact of applying different aggregators to the customized message propagation layer on the final experimental effect of the model. It can be seen in Table 4 that for a single aggregator, the effect of attention aggregation is the best, which is better than the average aggregation. The reason is very simple; attention aggregation can learn the importance of different neighbors. For cross-polymerization, a transformer (Trm) aggregator is better than an FM aggregator, which may be attributed to the coarse granularity of FM cross-aggregation. The best effect in the aggregator of the message propagation layer is attention + Trm + FM, which not only uses attention to aggregate the learning node representation but also uses FM to learn the second-order coarse-grained cross-information between neighbors and uses transformer to learn the fine-grained cross-information between neighbors.

Table 4. Effect of aggregators.

	Amazon-Book		**Last-FM**		**Yelp2018**	
	Recall	**Ndcg**	**Recall**	**Ndcg**	**Recall**	**Ndcg**
Mean	0.1452	0.0978	0.0854	0.1289	0.0698	0.0834
Attention (Att)	0.1480	0.1022	0.0890	0.1322	0.0720	0.0856
FM	0.1438	0.0965	0.0834	0.1284	0.0692	0.0812
Transformer (Trm)	0.1455	0.1016	0.0845	0.1275	0.0687	0.0820
Att + FM	0.1493	0.1077	0.0901	0.1340	0.0743	0.0893
Att + Trm	0.1522	0.1095	0.0912	0.1367	0.0750	0.0901
Att + Trm + FM	0.1571	0.1102	0.0930	0.1394	0.0775	0.0912

5.2.3. Ablation Study about the Improvement of ICKG

It can be seen in Figure 7 that the collaborative knowledge graph with user–user or item–item connection edges had a more significant effect than the original CKG, and the effect of adding user–user edges alone was better than adding item–item edges alone, reflecting the interest preference of modeling users and depicting the similarity between users, which is more effective than commodity similarity. The collaborative knowledge graph with user–user and item–item connection edges, at the same time, had the best effect, as shown in the red histogram above.

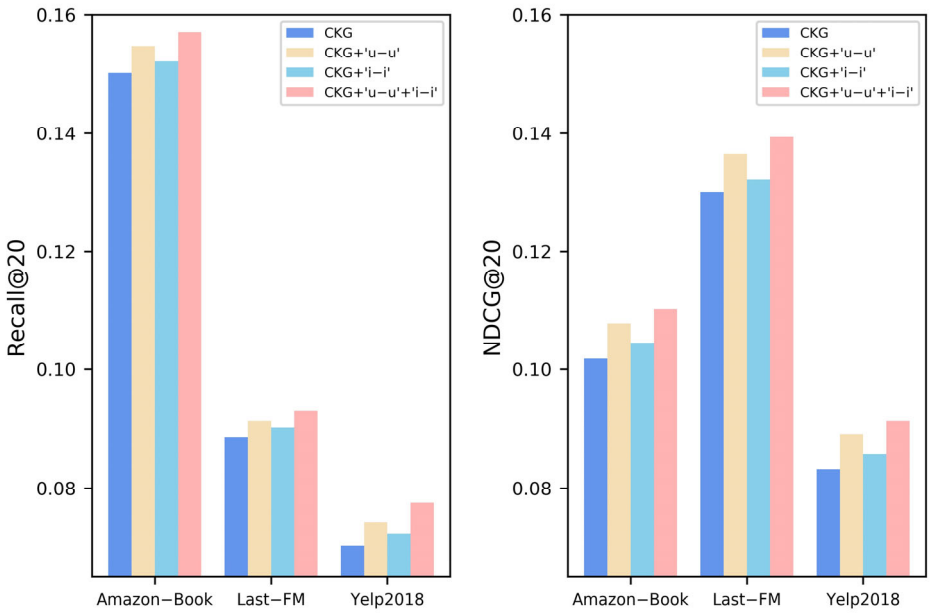

Figure 7. Experimental analysis of improved ablation of CKG.

5.2.4. Effect of the Number of Adding Edges

To explore the impact of connecting similar neighbors with different thresholds on the model effect in the process of constructing ICKG, we conducted the following experiments. We set K at {5, 10, 15, 20, 25, 30}. As can be seen in the experimental results in Table 5, a larger K did not indicate better performance. We conjecture the following reasons:

1. The smaller K is, the more edges are connected. However, noise is introduced, resulting in a decline in the model effect;
2. The larger K is, the fewer edges are connected. Therefore, some effective information between nodes is not connected, and the collaborative knowledge graph cannot be learned.

Table 5. Effect of the number of added edges.

K	Amazon-Book		Last-FM		Yelp2018	
	Recall	**Ndcg**	**Recall**	**Ndcg**	**Recall**	**Ndcg**
5	0.1520	0.1075	0.0905	0.1380	0.0734	0.0880
10	0.1532	0.1097	0.0912	0.1388	0.0750	0.0875
15	0.1554	0.1094	0.0921	**0.1401**	0.0766	0.0905
20	0.1571	**0.1102**	**0.0930**	0.1394	**0.0775**	**0.0912**
25	**0.1578**	0.1089	0.0912	0.1395	0.0765	0.0902
30	0.1544	0.1080	0.0905	0.1368	0.0754	0.0895

6. Conclusions

In this study, we committed to using a GNN knowledge graph to alleviate the feature sparsity and behavior sparsity in the recommender system. A multi-granularity enhanced graph neural network, MAKR, combining an attribute graph and interactive information was proposed. First, in order to enhance the high-order connectivity of graphs, based on the fusion of interaction graphs and attribute graphs, we proposed using similarity to connect similar users and items to form ICKG graphs. Second, in order to reduce the feature sparsity in the recommender system, we proposed a new aggregation method that used attention mechanisms, FM, and transformer as aggregators to aggregate neighbor information from different granularities. A large number of experiments on three real data sets proved the rationality and effectiveness of MAKR.

In this work, we explored the importance of high-order connectivity to the recommender system based on a knowledge graph, actively changed the aggregation mode of GNN in the past, and used comprehensive and detailed attention mechanisms, FM, and transformer to replace the common mean or maximum pooling method, which reduced the problem of heterogeneous information cancellation caused by the previous methods.

This paper did not consider tag records that could depict user profiles and item features. In the future, more side information, such as social relations and tag behaviors, could be considered in graph construction to improve the performances. In addition, we are also interested in exploring meta-path method in knowledge graph for recommendation.

Author Contributions: Conceptualization, X.L.; methodology, X.L.; software, X.L. and R.S.; validation, Y.W.; investigation, Y.W.; data curation, X.L.; writing—original draft preparation, X.L.; writing—review and editing, X.L. and R.S.; visualization, Y.W.; supervision, H.X. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (62077027), the Ministry of Science and Technology of the People's Republic of China(2018YFC2002500), the Jilin Province Development and Reform Commission, China (2019C053-1), the Education Department of Jilin Province, China (JJKH20200993K), the Department of Science and Technology of Jilin Province, China (20200801002GH), and the European Union's Horizon 2020 FET Proactive project "WeNet-The Internet of Us" (823783).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data underlying this article are available in the article.

Acknowledgments: The authors would like to thank all of anonymous reviewers and editors for their helpful suggestions for the improvement of this paper.

Conflicts of Interest: The authors certify that there is no conflict of interest in the subject matter discussed in this manuscript.

References

1. Warren, J.; Marz, N. *Big Data: Principles and Best Practices of Scalable Realtime Data Systems*; Simon and Schuster: Manhattan, NY, USA, 2015.
2. Feng, X.; Zeng, Y. A deep learning model of dynamic aspect attention ecommerce recommendation based on user comments. *Comput. Appl. Softw.* **2020**, *37*, 38–44.
3. Adomavicius, G.; Tuzhilin, A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng.* **2005**, *17*, 734–749.
4. Balabanovic', M.; Shoham, Y. Fab: Content-based, collaborative recommendation. *Commun. ACM* **1997**, *40*, 66–72.
5. Goldberg, D.; Nichols, D.; Oki, B.M.; Terry, D. Using collaborative filtering to weave an information tapestry. *Commun. ACM* **1992**, *35*, 61–70.
6. Basu, C.; Hirsh, H.; Cohen, W. Recommendation as classification: Using social and content-based information in recommendation. In Proceedings of the AAAI/IAAI, Madison, WI, USA, 27–29 July 1998; pp. 714–720.
7. Zhang, Z.K.; Liu, C.; Zhang, Y.C.; Zhou, T. Solving the cold-start problem in recommender systems with social tags. *Europhys. Lett.* **2010**, *92*, 28002.

8. Jamali, M.; Ester, M. A matrix factorization technique with trust propagation for recommendation in social networks. In Proceedings of the Fourth ACM Conference on Recommender Systems, Barcelona, Spain, 26–30 September 2010; pp. 135–142.
9. Zhen, Y.; Li, W.J.; Yeung, D.Y. TagiCoFi: Tag informed collaborative filtering. In Proceedings of the Third ACM Conference on Recommender Systems, New York, NY, USA, 22–25 October 2009; pp. 69–76.
10. 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.
11. Zhang, Y.; Ai, Q.; Chen, X.; Wang, P. Learning over knowledge-base embeddings for recommendation. *arXiv* **2018**, arXiv:1803.06540 2018.
12. Guo, Q.; Zhuang, F.; Qin, C.; Zhu, H.; Xie, X.; Xiong, H.; He, Q. *A Survey on Knowledge Graph-Based Recommender Systems*; IEEE: Piscataway, NJ, USA, CoRR 2020; abs/2003.00911.
13. 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.
14. 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.
15. 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.
16. 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.
17. 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.
18. 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.
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, 13–17 May 2019; pp. 3307–3313.
20. Ying, R.; He, R.; Chen, K.; Eksombatchai, P.; Hamilton, W.L.; Leskovec, J. Graph convolutional neural networks for web-scale recommender systems. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 974–983.
21. Perozzi, B.; Al-Rfou, R.; Skiena, S. Deepwalk: Online learning of social representations. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 24–27 August 2014, pp. 701–710.
22. Knudsen, E.I. Fundamental components of attention. *Annu. Rev. Neurosci.* **2007**, *30*, 57–78.
23. Rendle, S. Factorization machines. In Proceedings of the 2010 IEEE International Conference on Data Mining, Sydney, Australia, 13 December 2010; pp. 995–1000.
24. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*; doi： abs/1706.03762
25. Hamilton, W.L.; Ying, Z.; Leskovec, J. Inductive Representation Learning on Large Graphs. In Proceedings of the Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, Long Beach, CA, USA, 4–9 December, 2017; pp. 1024–1034.
26. Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; Bengio, Y. Graph Attention Networks. *arXiv* **2017**, arXiv:1710.10903.
27. Bollacker, K.D.; Evans, C.; Paritosh, P.K.; Sturge, T.; Taylor, J. Freebase: A collaboratively created graph database for structuring human knowledge. In Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, 10–12 June 2008; pp. 1247–1250.
28. 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.
29. Suchanek, F.M.; Kasneci, G.; Weikum, G. Yago: A core of semantic knowledge. In Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, AB, Canada, 8–12 May 2007; pp. 697–706.
30. Carlson, A.; Betteridge, J.; Kisiel, B.; Settles, B.; Hruschka, E.R.; Mitchell, T.M. Toward an Architecture for Never-Ending Language Learning. In Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, GA, USA, 11–15 July 2010.
31. Schlichtkrull, M.S.; Kipf, T.N.; Bloem, P.; van den Berg, R.; Titov, I.; Welling, M. Modeling Relational Data with Graph Convolutional Networks. In *Lecture Notes in Computer Science, Proceedings of the Semantic Web—15th International Conference, ESWC 2018, Heraklion, Greece, 3–7 June 2018*; Springer: Berlin/Heidelberg, Germany, 2018; Volume 10843, pp. 593–607.
32. Wang, X.; He, X.; Cao, Y.; Liu, M.; Chua, T.S. Kgat: Knowledge graph attention network for recommendation. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 950–958.

33. Lops, P.; de Gemmis, M.; Semeraro, G. Content-based Recommender Systems: State of the Art and Trends. In *Recommender Systems Handbook*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 73–105.
34. Nguyen, P.T.; Tomeo, P.; Noia, T.D.; Sciascio, E.D. Content-Based Recommendations via DBpedia and Freebase: A Case Study in the Music Domain. In *Lecture Notes in Computer Science, Proceedings, Part I, Proceedings of the The Semantic Web—ISWC 2015—14th International Semantic Web Conference, Bethlehem, PA, USA, 11–15 October 2015*; Springer: Berlin/Heidelberg, Germany, 2015; Volume 9366, pp. 605–621.
35. Pazzani, M.J.; Billsus, D. Content-Based Recommendation Systems. In *Lecture Notes in Computer Science, Proceedings of the Adaptive Web, Methods and Strategies of Web Personalization*, Berlin, Germany, 2007; pp. 325–341.
36. Yu, J.; Yin, H.; Li, J.; Wang, Q.; Hung, N.Q.V.; Zhang, X. Self-Supervised Multi-Channel Hypergraph Convolutional Network for Social Recommendation. In Proceedings of the WWW '21: The Web Conference 2021, Ljubljana, Slovenia, 19–23 April 2021; pp. 413–424.
37. Wu, L.; Li, J.; Sun, P.; Hong, R.; Ge, Y.; Wang, M. *DiffNet++: A Neural Influence and Interest Diffusion Network for Social Recommendation*; IEEE: Piscataway, NJ, USA, 2020; abs/2002.00844.
38. Wu, S.; Zhang, W.; Sun, F.; Cui, B. Graph Neural Networks in Recommender Systems: A Survey. *arXiv* **2020**, arXiv:2011.02260.
39. Wang, X.; He, X.; Wang, M.; Feng, F.; Chua, T. Neural Graph Collaborative Filtering. In Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, 21–25 July 2019; pp. 165–174.
40. He, X.; Deng, K.; Wang, X.; Li, Y.; Zhang, Y.; Wang, M. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2020, Xi'an, China, 25–30 July 2020; pp. 639–648.
41. Wu, S.; Tang, Y.; Zhu, Y.; Wang, L.; Xie, X.; Tan, T. Session-Based Recommendation with Graph Neural Networks. In *Proceedings of the Thirty-Third AAAI Conference on Artificial*; Palo Alto, CA, USA, 2019; pp. 346–353.
42. Mnih, A.; Hinton, G.E. A scalable hierarchical distributed language model. *Adv. Neural Inf. Processing Syst.* **2008**, *21*, 1081–1088.
43. Morin, F.; Bengio, Y. Hierarchical probabilistic neural network language model. In Proceedings of the International Workshop on Artificial Intelligence and Statistics, Bridgetown, Barbados, 6–8 January 2005; pp. 246–252.
44. Zhang, H.; Goodfellow, I.; Metaxas, D.; Odena, A. Self-attention generative adversarial networks. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 10–15 June 2019; pp. 7354–7363.
45. He, K.; Zhang, X.; Ren, S.; Sun, J. Identity mappings in deep residual networks. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 8–16 October 2016; pp. 630–645.
46. Ba, J.L.; Kiros, J.R.; Hinton, G.E. Layer normalization. *arXiv* **2016**, arXiv:1607.06450 2016.
47. Xu, K.; Li, C.; Tian, Y.; Sonobe, T.; Kawarabayashi, K.I.; Jegelka, S. Representation learning on graphs with jumping knowledge networks. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 5453–5462.
48. Rendle, S.; Freudenthaler, C.; Gantner, Z.; Schmidt-Thieme, L. BPR: Bayesian personalized ranking from implicit feedback. *arXiv* **2012**, arXiv:1205.2618 2012.
49. He, X.; Chua, T.S. Neural factorization machines for sparse predictive analytics. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Tokyo, Japan, 7–11 August 2017; pp. 355–364.
50. Ai, Q.; Azizi, V.; Chen, X.; Zhang, Y. Learning heterogeneous knowledge base embeddings for explainable recommendation. *Algorithms* **2018**, *11*, 137.
51. Berg, R.V.D.; Kipf, T.N.; Welling, M. Graph convolutional matrix completion. *arXiv* **2017**, arXiv:1706.02263 2017.
52. He, X.; Liao, L.; Zhang, H.; Nie, L.; Chua, T.S. Neural Collaborative Filtering. In Proceedings of the 26th International Conference, Perth, Australia, 3-7 April 2017; pp. 173-182.
53. Yang, J.H.; Chen, C.M.; Wang, C.J.; Tsai, M.F. HOP-rec: High-order proximity for implicit recommendation. In Proceedings of the 12th ACM Conference on Recommender Systems, Vancouver, BC, Canada, 2–7 October 2018; pp. 140–144.