entropy

Article

KHGCN: Knowledge-Enhanced Recommendation with Hierarchical Graph Capsule Network

Fukun Chen †, Guisheng Yin †, Yuxin Dong *, Gesu Li and Weiqi Zhang

School of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China
* Correspondence: dongyuxin@hrbeu.edu.cn
† These authors contributed equally to this work.

Abstract: Knowledge graphs as external information has become one of the mainstream directions of current recommendation systems. Various knowledge-graph-representation methods have been proposed to promote the development of knowledge graphs in related fields. Knowledge-graph-embedding methods can learn entity information and complex relationships between the entities in knowledge graphs. Furthermore, recently proposed graph neural networks can learn higher-order representations of entities and relationships in knowledge graphs. Therefore, the complete presentation in the knowledge graph enriches the item information and alleviates the cold start of the recommendation process and too-sparse data. However, the knowledge graph's entire entity and relation representation in personalized recommendation tasks will introduce unnecessary noise information for different users. To learn the entity-relationship presentation in the knowledge graph while effectively removing noise information, we innovatively propose a model named *knowledge—enhanced hierarchical graph capsule network* (KHGCN), which can extract node embeddings in graphs while learning the hierarchical structure of graphs. Our model eliminates noisy entities and relationship representations in the knowledge graph by the entity disentangling for the recommendation and introduces the attentive mechanism to strengthen the knowledge-graph aggregation. Our model learns the presentation of entity relationships by an original graph capsule network. The capsule neural networks represent the structured information between the entities more completely. We validate the proposed model on real-world datasets, and the validation results demonstrate the model's effectiveness.

Keywords: knowledge graph; recommendation system; graph neural network; attention mechanism

check for updates

Citation: Chen, F.; Yin, G.; Dong, Y.; Li, G.; Zhang, W. KHGCN: Knowledge-Enhanced Recommendation with Hierarchical Graph Capsule Network. *Entropy* **2023**, *25*, 697. https://doi.org/10.3390/e25040697

Academic Editors: Andrea Prati, Luis Javier García Villalba and Vincent A. Cicirello

Received: 27 March 2023
Revised: 13 April 2023
Accepted: 17 April 2023
Published: 20 April 2023

1. Introduction

Recommendation systems have shown great potential in solving various online applications' rapidly growing information volume problems, improving user efficiency, and increasing stickiness. In commercial platforms, such as Netflix and Amazon, recommendation systems provide excellent convenience for users by filtering out target items from thousands of movies and TV shows or millions of products [1,2]. To cope with the problems caused by the ever-increasing data, researchers have also developed neural network models and deep learning methods that process massive amounts of information. Deep neural networks have considerable advantages in many information-dominated fields, such as healthcare [3] and the Internet of Things (IoT) [4], especially in recommendation systems that rely on data [5,6].

Compared with traditional machine learning methods, deep learning methods can more effectively incorporate additional information (such as text, pictures, and so on) related to the recommendation task. Data-sparse and cold-start problems in recommendation systems (such as collaborative filtering-based recommendation systems) only utilizing user–item interactions for recommendation usually face a sharp drop in recommendation performance during user–user or user–item interactions. To alleviate the troubling data-sparse and cold-start problems in collaborative filtering (CF) based recommendation systems, researchers usually add the side information of users and items and design sophisticated algorithms to utilize this information. Various types of auxiliary information have been utilized to alleviate these problems.

Researchers use valuable external knowledge as extra information, such as reviews, social networks, tags, item attributes, etc., to improve the effectiveness of recommendations. Wang et al. used neural networks to extract the embedding of ratings and reviews, respectively. Further, they combined the embedding with a collaborative filtering method to propose a hybrid deep collaborative filtering model [7]. Chen et al. introduced label information into the recommendation system and proposed a label intersection model by studying the intersections between user labels and item labels for a better recommendation [8]. Shi et al. were motivated to utilize reviews and further reduced the dual graph convolutional network method to capture the full description of an aspect in all reviews for the recommendation [9]. The knowledge graph (KG) can encode users, items, and attributes related to items in the graph structure to preserve relationship information, thus attracting a wide range of research interests. Recently, researchers have explored the recommendation system based on the knowledge graph [10–14]. Knowledge-graph-embedding (KGE) methods integrate the KG at the recommendation system, knowledge-aware recommendation, to advance the accuracy and interpretability of the recommendation task, which has catalyzed considerable research works [15–18]. Researchers consider KG effective for improving quality recommendations because user and item attributes in KG are essential auxiliary information. Integrated interactions between the user and item and the attributes in KG (that is, they appear in the sample data simultaneously) can significantly improve the prediction accuracy in various recommendation systems.

Figure 1 depicts an example of a collaboration knowledge graph (CKG) in the film domain [19] to infer the preference of users u_4 and u_5 for item i_5.As shown in the user–item interactions section in Figure 1, the CF-based method determines whether to recommend or not by calculating the value of the degree of association (cosine similarity, etc.) among the u_4, u_5, and i_5. For example, for $\text{sim}(u_4, i_5) = \text{sim}(u_5, i_5) = 0$, the result is negative. In the knowledge graph part of Figure 1, item i_3 (preferred by user u_4), item i_4 (preferred by user u_5), and item i_5 have the same attribute e_3. We refer to the hybrid structure of the knowledge graph and user–item graph as CKG, which takes the user–item interactions and knowledge graph into account. According to the attribute e_3 in CKG, there is the interaction between users u_4, u_5, and movie i_5, which is connected by relation Gener. Obviously, at this time, the recommendation system can mine the user's more prolonged, deeper, and more profound preference items to enhance the recommendation performance and interpretability of the recommendation system. Next, the latest GNN-based algorithms can further mine higher-order relationships between the user–item entities in CKG. However, to exploit higher-order information in the CKG graph, the following questions are noteworthy:

1. With the increase in high-order size, the number of nodes related to the target user increases sharply, which increases the computational load of the model.
2. Although the number of nodes related to target users has increased, nodes under higher-order relationships have different effects on the recommendation. Therefore, the contribution of nodes requires further screening by algorithmic models.
3. The increase in nodes is a double-edged sword. Knowledge graphs inevitably introduce specific noise. Therefore, it is difficult for previous research methods to characterize target users and candidate items to generate an accurate recommendation list. Accurate user-embedding learning is essential for modern recommendation systems.

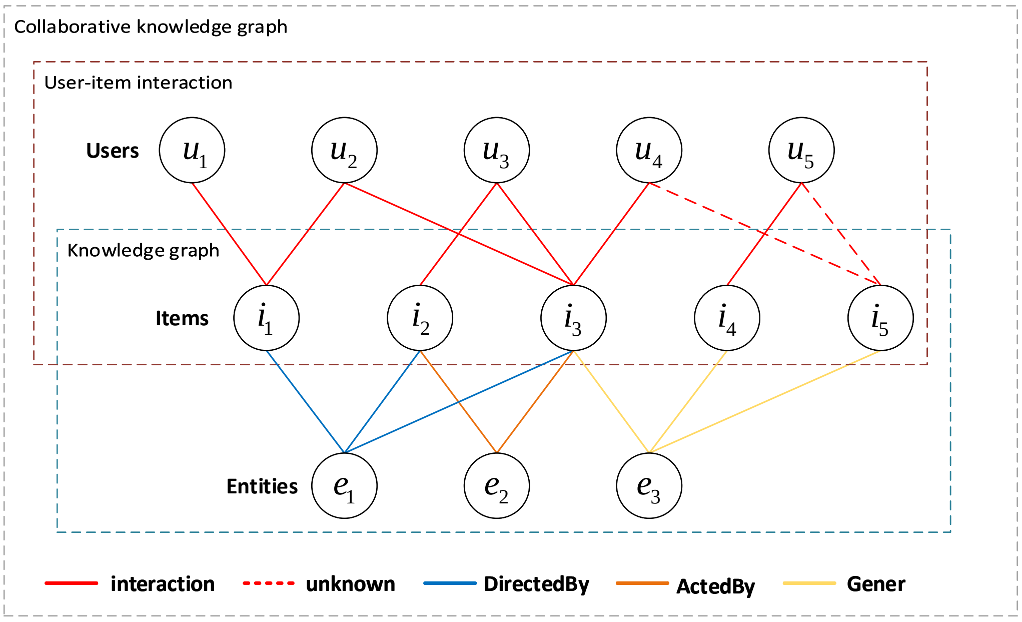

Figure 1. A toy example of the CKG contains users, movies, entities, interactions, unknown, directed by, and gener as relations.

Graph structure information provides valuable guidance for graph neural networks learning node representation [20,21]. The excellent potential of the graph convolutional network (GCN) for recommending each is due to its ability to learn better embeddings of the user and item by using collaborative representations from high-level neighbor nodes [22,23]. Usually, attributes are not isolated but interconnected, which forms a KG. One must go beyond the obligation to model user–item interactions and consider auxiliary information to provide more accurate, diverse, and interpretable recommendations. Recently, leveraging knowledge graphs in recommendation systems to alleviate data-sparse and cold-start problems has attracted considerable attention. Despite recent advances in graph neural networks (GNNs), high-level collaboration signals are combined to alleviate the problem. Still, the ending of cold-start users and items is not optimized, and cold-start neighbors are not handled during startup. Solving the cold-start problem in recommendation systems is crucial for new users and new items. Under the premise of sparsely observed data, how to further mine unobserved user–item pairs is also an important research direction to refine users' potential preferences. The other social relationships are usually used to improve the recommendation quality when considering the sparsity of the user–item interactions in the social recommendation.

Graph neural networks (GNNs) have significant advantages in the display modeling of structured data. However, existing GNNs are limited in their ability to capture the representation of the hierarchy, and the hierarchy graph plays an important role in the representation learning of the graph. Like other GNN models, GNN-based recommendation models inevitably suffer from the problem of over-smoothing. Over-smoothing is when the graph neural network stacks more layers, and the node embeddings in the network become increasingly similar until they become indistinguishable, resulting in performance degradation. Aiming at the problems of graph representation learning, researchers try to use the multi-channel feature of capsGNN to learn a complete graph structure from the perspective of breadth [24–28]. In graph representation learning, capsule GNN has also achieved good results [29]. Xu et al. proposed a taxonomy-enhanced graph neural network (Taxo-GNN), which jointly optimizes the taxonomy representation and node representation tasks, where categories in taxonomy are mapped to Gaussian distributions and nodes are embedded with the GNN framework [30]. Inspired by the capsule neural network (CapsNet) [31], we propose a hierarchical graph capsule network (HGCN), which uses the capsule concept to solve the shortcomings of existing GNN-based graph-embedding algorithms. By extracting node features in the form of capsules, the routing mechanism can be used to capture important information in the hierarchy graph. However, there are noisy nodes in the graph structure constructed by the knowledge graph. Most GNN-based recommendations will also learn the representation of the noisy node when learning the presentation of graph structure information. To alleviate the restrictions of the recommendation methods with KG and GNN, we propose a novel transformer graph attention network (TGAT) component in our model for high-order information propagation in the collaborative knowledge graph. The attentive mechanism in TGAT is weighted to judge the importance of each entity; the attention mechanism is used to extract the significance of each channel; and multiple screenings weaken the significance of noisy entities and strengthen the reputation of crucial entities. We summarize the problems existing in the recommendation algorithms with KG and GNN. From the perspective of a knowledge graph, there are noisy entities and relationships in a complete knowledge graph. From the perspective of graph neural network methods, graph neural networks, such as GCN itself, face the problem of model bottlenecks. To solve the problems that CKG and graph neural networks face in recommendation systems, we propose the knowledge-enhanced hierarchical graph capsule network method, which alleviates the problems existing in the incorrect forms. In general, the contribution of our model is as follows:

1. Node disentangle is introduced, and the problem of the representation of noisy entities and relationships in the complete KG for recommendations is alleviated, generating a disentangled user–item knowledge graph;
2. A novel attention scoring function is designed to more effectively aggregate the nodes in the disentangled knowledge graph and create a more accurate representation of the user and item;
3. The introduction of KHGCN, the use of the hierarchical transformer graph attention network (TGAT) to enhance the ability to represent relationships and entities in the graph, learn the representation of entity relationships in the KG more efficiently, and make accurate recommendations;
4. An end-to-end model recommendation framework is constructed, surpassing existing state-of-the-art methods on three real-world datasets and four evaluation indicators.

2. Related Work

2.1. Knowledge Graph for Recommendations

Introducing the knowledge graph for the recommendation system can effectively alleviate the data-sparse and cold-start problems. In recent years, many researchers have been performing related work. To model both the dynamic interests and the dynamic social influences, Gu et al. proposed a method to model and integrate item-embedding representations and contextual friendship representations for recommendation tasks [32].

To integrate the knowledge graph with the recommendation system under a unified framework, researchers combine the knowledge graph's feature learning with the recommendation algorithm's objective function and train in the same end-to-end objective process. Zhang et al. exploited heterogeneous information in the knowledge graph to enhance the performance of recommendation systems. They proposed that the integrated framework of collaborative knowledge-graph embedding is learning semantic representations in items from the knowledge graph while learning latent representations of methods in collaborative filtering [33]. Dong et al. proposed a novel and simple model to achieve a low computational cost, which adopts a semi-automatic encoder to embed item attributes and graph features simultaneously for recommendations [34]. In [12], a hybrid recommendation system based on attention mechanism and knowledge-graph embedding (HRS) is put forward to alleviate sparsity and cold-start problems and improve the performance of the recommender systems. Zou et al. focused on exploring the contrastive learning in KG-aware recommendation and proposed a novel multi-level cross-view contrastive learning mechanism (MCCLK), which comprehensively considers three different graph views for KG-aware recommendation, including the global-level structural view, local-level collaborative and semantic views [10]. In [14], Zou et al. focused on exploring contrastive learning in KGR and proposed a novel multi-level interactive contrastive learning mechanism, which contrasts nodes of two generated graph views. To tackle the problem that most existing KG-based recommender systems ignore the fact that users attach different degrees of importance to various relationships of items, Zhang et al. proposed a knowledge-graph recommender model based on adaptive relational attention (KGARA) [11]. Yin et al. proposed a model for knowledge-aware recommendation systems, where the main components are an attribute regularizer and a dynamic attention mechanism, which are attention components. They explored potential connections between the users and items to enhance the recommendation performance [35].

Since both recommendation and knowledge-graph embedding tasks are included, knowledge-based recommendation systems are similar to multi-task learning frameworks, which are related by items and entities in the knowledge graph. Knowledge-graph representation and recommendation tasks are inevitably related. The knowledge-graph link prediction can assist the recommendation system to keep away from the local minimum and prevent the recommendation system from overfitting, thereby improving generalization. Some researchers regard recommendation systems and knowledge-graph feature learning separately and only adopt a multi-task learning framework in the final loss function training. Wang et al. proposed a compression unit to correlate the two tasks, which automatically shares latent features while also learning higher-order interactions between the user–item interactions and knowledge-graph entities [36]. The structured data features of knowledge graphs have unique advantages compared with other auxiliary information in personalized recommendation systems, which enhance the interpretability of the recommendations' knowledge-based representation learning framework to embed heterogeneous entities for the recommendation based on the knowledge-graph embedding, and a soft matching algorithm is proposed. Regarding personalized explanations for the recommended items [37], Liu et al. proposed a knowledge-graph-enhanced multi-task learning method to learn cross features from ratings and reviews by fusing users and movies with their review knowledge entities in the same graph [13].

2.2. GNN for Recommendations

With the deepening of knowledge-based recommendation-systems research, the relational structure of external knowledge can be introduced while learning user and item representations. In this regard, traditional neural networks have successfully extracted features from Euclidean spatial data but, in many cases, still contain a lot of non-Euclidean spatial data. However, the performance of traditional neural networks is still unsatisfactory when addressing non-Euclidean spatial data. As the performance is still unsatisfactory, the researchers draw on the ideas of convolutional neural networks, recurrent neural networks, and deep auto-encoders to define the graph neural networks that can process non-Euclidean spatial data. Since then, the related research of graph neural networks has sprung up as new hot research.

In recent years, researchers have used GNN for recommendations with great success. Thomas N. Kipf and Max Welling proposed a graph convolutional new algorithm derived from the idea of the convolution algorithm and able to be directly used to process structural graph data [20]. Another feature of GCN is that its model size increases linearly with the number of edges in the graph. Given the problems of GCN addressing large-scale graphs, GraphSAGE is proposed for inductive representation on large graphs. GraphSAGE is a framework for inductive representation learning on large graphs, which is used to generate low-dimensional vector representations of nodes that can efficiently process graphs with many nodes' attribute information [20]. For the fine-grained division of node attributes information in a graph, Petar et al. proposed new neural network architecture, the graph attention network (GAT), which learns the node weights in the graph structure through a masked self-attention layer [21]. It effectively alleviates the shortcomings of other methods, such as GCN.

Some researchers have studied recommendations based on graph neural networks, among which the works based on the GCN framework are as follows. In [38], Chen et al. proposed a GCN-based linear collaborative model with a residual network component, which can relieve the over-smoothing problem in higher-order graph convolution to a certain extent. Furthermore, He et al., through empirical observation, found that the two components of feature transformation and nonlinear activation in the structure of GCNs do not contribute much to improving collaborative filtering. A new model named LightGCN based on GCN is proposed, and the fundamental neighbor aggregation part in GCN is reserved for collaborative filtering [23]. In [39], the problem of hashing with GNN for high-quality retrieval was investigated, and a deep hashing with GNN framework was proposed to learn continuous and GNN codes jointly. Wu et al. proposed an adaptive GCN method for knowledge-based recommendations, effectively integrating the item recommendations and attribute inference [40]. The attribute inference part in adaptive GCN can adjust the graph-embedding learning parameters and refine the item attributes and preference behavior through the weak supervision information provided by the attribute inference part to complete the item recommendation. Previous studies failed to model the various impacts of multi-behavior in multi-behavior recommendations. In [41], a new model, multi-behavior GCN, was proposed to solve the problems in multi-action recommendation tasks. The main idea of multi-behavior GCN is to construct a multi-behavior graph and then learn the representation of the multi-behavior graph through an improved graph neural network. To solve food-recommendation problems, Gao et al. proposed an innovative graph convolutional network, which hierarchically learns the relationships in the food graph and models higher-order relationships in the food graph through an information propagation mechanism [42]. To solve oscillation problems, Liu et al. proposed a new model for recommendation tasks, named deoscillated adaptive graph collaborative filtering, which includes the layer-wise propagation patterns and can adaptively learn local factors [43]. To learn the fine-grained property features, Ge et al. proposed an end-to-end model called the collaborative property-aware graph convolutional network (CPGCN), which fuses the user–service collaborative information with the semantic information of KG to construct collaborative property-aware graphs (CPGs) [44]. Wang et al. proposed a new knowledge graph-aware light graph convolutional network (KLGCN), which removes feature transformation and nonlinear activation in KG-aware recommendation while improving performance [45]. Zhu et al. embedded users' interests from their social information by the attentional graph convolutional network (GCN) and improved news representations via attention mechanisms [46].

The works based on the GraphSage framework are as follows. In [47], a matrix completion model based on the inductive graph is proposed, which performs 1-hop subgraph expansion through the user–item interaction graph generated based on the rating matrix, and aggregates these subgraph representations through GNN. Sun et al. proposed a new framework that can explicitly aggregate neighbor node information, efficiently modeling user–item bipartite heterogeneous graphs [48]. In the online recommendation services, Xu et al. developed a framework based on incremental learning graph neural networks to address the catastrophic forgetting problem faced during incremental learning training, which implements a graph structure preservation strategy to preserve users' long-term preferences when the model is updated [49]. In the personalized video highlight recommendations, Wu et al. proposed an inductive transfer learning framework consisting of a graph neural network and an item embedding transfer network for the personalized recommendations of video highlights [50].

The works based on the GAT framework are as follows. To capture fine-grained user preference, in [51], an inductive transfer learning framework was proposed, which consists of a graph neural network and an item embedding transfer network for the personalized recommendations of video highlights. Regarding social recommendations, Wu et al. proposed a method (DiffNet++) based on the DiffNet to combine social networks and interest networks in social recommendations to construct a new social interest heterogeneous graph to redefine social recommendations [52]. Furthermore, DiffNet++ injects user–user high-order implicit feedback in the social network, injects user–item high-order implicit feedback in the interest network, and infiltrates the user-embedding representation in a fine-grained manner. To effectively learn the contribution of different neighbor entities to the target embedded in the process of neighborhood propagation, they developed a hierarchical attention mechanism at the neighbor level and graph level, which can adaptively learn important content at different levels, and provides more possibilities for social recommendations [53]. In multimodal recommendations, Sun et al. proposed multi-modal knowledge GAT, a multimodal graph attention technology that uses a graph attention network to model multimodal information on a multimodal knowledge graph [54]. The modal knowledge-graph attention mechanism integrates the text and image information of entities to the tail node of the knowledge graph, which enriches the entity representation of the knowledge graph. Feng et al. proposed a model that can effectively capture user–item pairs in knowledge graphs and employ graph-connection and graph-pruning techniques to construct the behavioral graphs of adaptive targets in collaborative knowledge graphs [55]. Wang et al. proposed a new method called knowledge GAT (KGAT), which innovatively handles knowledge graphs with graph attention neural networks and proposes a new neighbor aggregation method to improve the target node's embedding; the attention mechanism in KGAT can effectively divide the importance of neighbor nodes [19].

3. Models and Methods of KHGCN for Recommendation

This section discusses in detail the knowledge-enhanced hierarchical graph capsule network (KHGCN) proposed to address the problems of knowledge-based recommendation tasks. KHGCN aims to learn node embeddings in collaborative knowledge graphs as well as fine-grained learning of the representation of target nodes in the CKG using a hierarchical graph capsule neural network. We take the dissociated representation of nodes in a collaborative knowledge graph as input to a graph capsule neural network. Therefore, the primary capsule in each graph capsule neural network is composed of multiple disentangled independent latent factors, where each latent factor represents a different attribute of the entity. Transformer GATs (TGATs) generate the primary capsules, whose main function is to encode the part–whole relationship between the low level and high level by further integrating the high-order neighbor representations of target nodes in collaborative knowledge graphs. Specifically, the instantiation parameters in the upper-layer capsules are obtained by the voting of the lower-layer capsules TGAT, which depends on the structure between the lower-layer capsules. The lower layer is routed to high-level capsules through the routing mechanism of the capsule neural networks. After several rounds of iterations, high-order representations of user items are obtained through objective function training. Finally, the obtained user and item representations are fitted by the prediction layer to complete the recommendation. Figure 2 shows the description of the KHGCN model framework and other components.

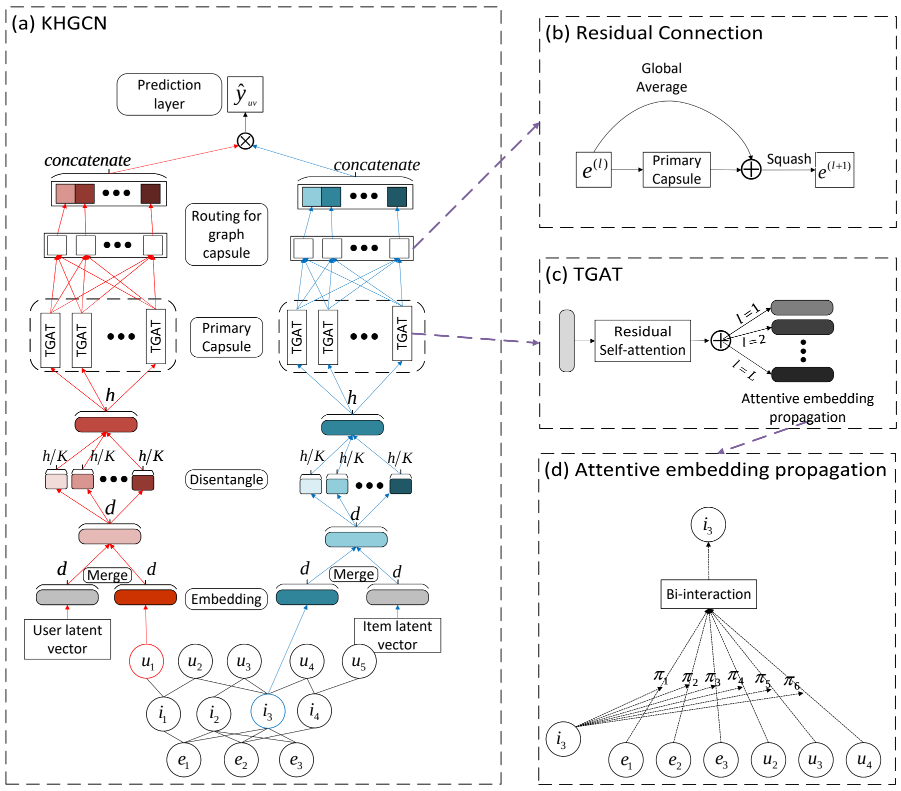

Figure 2. This is a figure. Schemes follow the same formatting. If there are multiple panels, they should be listed as follows: (**a**) The framework of KHGCN. (**b**) The residual connection in the routing part. (**c**) The TGAT layer part in the primary graph capsule. (**d**) The attentive embedding propagation layer part in the TGAT.

3.1. Problem Formulation

In this section, we introduce a detailed definition of the KHGCN model for the CKG-based recommendation. According to the collaborative knowledge graph constructed in Figure 1, the datasets are divided into the following three parts for introduction. User–item interaction graph: This contains users, items, and user–item interactions. The set of users is denoted as $U = \left\{u_1, u_2, \cdots, u_{|U|}\right\}$. The set of items is denoted as $V = \{v_1, v_2, \cdots, v_{|V|}\}$. Moreover, the user–item interactions set is denoted as $Y = \{y_{uv} | u \in U, v \in V\}$. For each interaction, $y_{uv} = 1$ means that the user u has positive feedback for item v, and $y_{uv} = 0$ means that the user u has no feedback for item v.

Knowledge graph: The knowledge graph consists of entity connection relationships. The entity set is denoted as $E = \{e_1, e_2, \cdots, e_{|E|}\}$, and $|E|$ represents the number of entities in knowledge graph. The relationship set is denoted as $R = \{r_1, r_2, \cdots, r_{|R|}\}$, and $|R|$ represents the number of relations in knowledge graph. where (h, r, t) is a triple in $\mathcal{G}$, and h, t represent the entities of head and tail, and relation r represents the relation between the head h and tail t. Here, we adopt a set of triples to represent the entire knowledge graph $\mathcal{G}(E, R, E)$.

Collaborative knowledge graph: This is a joint user–item graph and knowledge graph, for which we define user–item interactions as $\hat{y}_{uv} = \mathcal{F}(u; v; \Theta, Y, \mathcal{G})$, where $\hat{y}_{uv}$ represents the predicted interaction between the user u and item v pairs in the collaborative knowledge graph, and Θ is the parameter of the function $\mathcal{F}$.

Candidate items set for the target user are generated according to the predicted interactions. Some of them are selected for the recommendation list according to the actual situation. The function $\mathcal{F}$ is defined by the KHGCN. KHGCN generates the top@N list for the CKG-based recommendation, where N is the number of items in the recommendation list.

Figure 2a shows the overall framework of the recommendation model KHGCN, where the main components of the model are the embedding of CKG, disentanglement, capsule neural networks, and hierarchical graph attention networks.

3.2. Knowledge Graph Embedding

Knowledge-graph embedding maps entities and relations in a knowledge graph to a unified space for efficient representation, while preserving the structure of the knowledge graph. KHGCN adopts TransR [17], a widely used KGE method, to learn representations of knowledge graphs. Specifically, TransR learns the translation principle for (h, r, t) triples and learns to embed each triple existing in the knowledge graph by optimizing the translation principle $e_h^r + e_r \approx e_t^r$. Where $e_h, e_t \in R^d$ and $e_r \in R^k$, d are the embedding dimensions of the entity in the knowledge graph, k is the embedding dimension of the relation in the knowledge graph, and e_h^r and e_t^r are the mappings of the entities h and t in the relation r space. To sum up, given the triples present in the knowledge graph, the energy score formula calculated by TransR is as follows:

$$g(h, r, t) = ||W_r e_h + e_r - W_r e_t||_2^2 \tag{1}$$

where $W_r \in \mathbb{R}^{k \times d}$ is the spatial transformation matrix of relation r, and the transformation matrix is used to map d- dimensional entities into k-dimensional relation space r. In the formula, if the triplet has a higher score, the triplet is likely to be not true, and the lower the triplet score, the more likely it is to be true.

3.3. Node Disentangle

In most cases, highly complex interactions are involved in connecting each node pair in a graph. For example, the researchers largely model the relationships uniformly as before while neglecting the diversity of user intentions in watching the films, which could be for the director, actors, accompanying family or friends, etc. Therefore, it is necessary to disentangle the interpretable latent factors underlying the variation of entity representations in knowledge graphs. We expect to completely dissociate the independence between different intents in entity representations [29,43].

Motivated by [31], we add the disentanglement component into our model, which can be called disentangled HGCN, to disentangle the entities' representation factors and focus on disentangled representations for recommendation tasks. The dissociated entity node is used as the input of the graph capsule. Each graph capsule is composed of multiple independent heterogeneous factors, where each factor describes a piece of the representation of the entity.

For a knowledge graph, $G = (E, R, E)$, node i in the graph is represented by $e_i \in \mathbb{R}^d$. Specifically, we project the input entity features into K different subspaces, assuming there are K latent factor parameters:

$$e_{i,k} = \sigma(W_k^T e_i) + b_k \tag{2}$$

where $W_k \in \mathbb{R}^{d \times \frac{d_h}{K}}$, and $b_k \in \mathbb{R}^{\frac{d_h}{K}}$ are learnable parameters, σ is a nonlinear activation function, and $\frac{d_h}{K}$ is the dimension of each factor. Our study uses linear projection due to its efficiency and remarkable performance. Therefore, each graph capsule is represented by a pose matrix $E_i \in \mathbb{R}^{K \times \frac{d_h}{K}}$ [56].

After the dissociation operation, we reshape E_i into the dissociated vector format $E_i \in \mathbb{R}^{d_h}$. At this point, we obtain the de-dissociated entity vector to match the capsule graph's input. Therefore, we compress z_i as follows [31]:

$$p_i = squash(e_i) = \frac{||e_i||^2}{1 + ||e_i||^2} \frac{e_i}{||e_i||} \tag{3}$$

where $e_i^{(1)} = p_i \in \mathbb{R}^{d_h}$ is the primary graph capsule representing the lowest-level entity and the basis upon which all advanced capsules are founded.

3.4. Hierarchical Graph Capsule Learning

We propose the hierarchical graph capsule layers, consisting of TGATs and residual routing parts. The task of TGATs is to vote for the instantiation parameters of higher-level graph capsules.

Figure 2c shows the attentive embedding propagation layer part. Unlike KGAT [19], we pass the dissociated vector through a layer of the self-attention mechanism with residual values and then feed the resulting vector into the improved attention mechanism and embed the propagation layer.

3.4.1. Transformer Graph Attention Layers

- **Residual Transformer**: As shown in Figure 2b, we add a residual self-attention mechanism before the attention embedding propagation, which facilitates better coarse extraction of the embedding representation of the graph:

$$f_{RT}(X) = (\lambda_t + softmax(\frac{Q_t {K_t}^T}{\sqrt{d_{K_t}}})V_t)X \tag{4}$$

  where $Q_t = K_t = V_t = X$, $\lambda_t = 1$, and $X = e_i^{(1)}$ is the representation of disentangled entity embedding and relation in knowledge-enhanced graph G. The embedding of the i-th node $e_i^{(1)}$ can be represented as $e_i^{(1)} = f_{RT}(e_i^{(1)})$, where RT is short for residual-transformer.
- **Knowledge-aware graph attention**: For each entity h in graph G, we define its associated neighbor entity as $N_h = (h,r,t)|(h,r,t) \in G$, where N_h represents the set of triples associated with the head node h. The attentive embedding propagation layer part is shown in Figure 2d.
  To characterize the first-order connectivity structure of entity h, we obtain the relevant first-order neighbor embedding by computing the relation score π and the relevant entity t in N_h as

$$e_{N_h} = \sum_{(h,r,t) \in (N_h)} \pi(h,r,t) e_t \tag{5}$$

  where $\pi(h,r,t)$ controls the decay factor on each propagation on edge (h,r,t), indicating how much information is being propagated from t to h conditioned to relation r. In this part, we construct a new formula, and the learned node embedding with attention is integrated with the bi-interaction function. The improved formula for calculating $\pi(h,r,t)$ is as follows:

$$\pi(h,r,t) = (\tanh(W_r e_t + e_r))^T \tanh(W_r e_h + e_r) \tag{6}$$

  where we select $tanh$ as the nonlinear activation function. The attention score $\pi(h,r,t)$ between nodes h and t is determined by the relational spatial distance of relation r in TransR. The higher the degree of association to node h in the relationship space, the higher the value of the attention score $\pi(h,r,t)$. After calculating the scores of all nodes t related to node h in N_h, we normalize the attention scores by the softmax function:

$$\pi(h,r,t) = \frac{exp(\pi(h,r,t)}{\sum_{(h,r',t') \in N_h} exp(\pi(h,r',t'))} \tag{7}$$

  Therefore, the final attention score determines that nodes with high scores should be given more weight and attention so that high-order features in the neighbor set of the target node can be learned more accurately. During the training process, the attention scores divide the importance of each datum while enhancing the interpretability of the recommendation.

- **Bi-interaction aggregator**: The Bi-interaction aggregator is the result of considering various feature interactions between e_h and e_{N_h}. The Bi-interaction calculation formula is as follows:

$$f_{Bi} = LeakyRelu(W_1(e_h + e_{N_h}) + LeakyRelu(W_2(e_h \odot e_{N_h})) \tag{8}$$

where W_1, and $W_2 \in \mathbb{R}^{d' \times d_h}$ are the trainable weight matrices, and $\odot$ denotes the element-wise product. We encode the feature interactions between e_h and e_{N_h}. This term makes the information being propagated sensitive to the affinity between e_h and e_{N_h}, e.g., passing more messages from similar entities, where Bi is short for Bi-interaction. We encode the target node h embedding and its corresponding neighbors' set N_h embedding by Bi. Through the encoding method of Bi, we can further encode the high-order connection of the target node in the graph. Our method is built upon TGATs by following the attention information propagation paradigm:

$$e_h^{(l+1)} = f_{Bi}(e_h^{(l)}, e_{N_h}^{(l)}) \tag{9}$$

The final representation of node e_h is given by TGAT($e_h^{(1)}$, $e_{N_h}^{(1)}$) = $e_h^{(L)} \in \mathbb{R}^{L \times d_h}$ after L iterations.

3.4.2. Learning Primary Capsules

For all primary capsules in layer l, generated by the proposed hierarchical TGATs, it is assumed that the m-th node embedding of layer l can be calculated via

$$p_j^{(l)} = TGAT_j(e_i^{(l)}, e_{N_i}^{(l)}) \tag{10}$$

After stacking the hierarchical TAGTs outputs, we concatenate the primary capsules of node i at layer l into a vector $p^{(l)}$. The j-th primary capsule is represented as $p_j^{(l)}$, where $p_j^{(l)} \in \mathbb{R}^{L \times d_h}$.

3.4.3. Routing for Graph Capsules

After the primary capsule is generated, the obtained primary capsule generates the graph capsule through the graph routing mechanism. To iteratively generate the graph capsules, in each iteration, there are

$$e_i^{(l+1)} = squash(\sum_i c_{i,j}^{(l)} p_{j|i}^{(l)}) \tag{11}$$

where $c_{i,j}^{(l)} \geq 0$, and $\sum_{i=1}^{L} c_{i,j}^{(l)} = 1$. $q_j^{(l+1)}$ is the j-th graph capsule in layer $l+1$, representing the close voting cluster from the primary capsules in layer $l+1$, and $c_{i,j}^{(l+1)}$ is the routing coefficient used to calculate the voting of each primary capsule for the graph capsule, which represents the primary capsule $p_j^{(l)}$ relative to the graph capsule $e^{(l+1)}$.Here, consider the importance of $c_{i,j}^{(l)}$, which is iteratively updated using a graph routing mechanism.

The vote of each capsule in the primary capsules $p^{(l)}$ is routed to a capsule in the graph capsule $e^{(l+1)}$, and each graph capsule has the participation of the primary capsule. Formally, the vote routing coefficient $c_{i,j}^{(l)}$ is defined by softmax:

$$c_{i,j}^{(l)} = exp(b_{i,j}^{(l)}) / \sum_s exp(b_{i,s}^{(l)}) \tag{12}$$

where the initial value of $b_{i,j}^{(l)}$ is defined as 0. We perform R iterations of the graph-routing mechanism at each iteration, and $b_{i,j}^{(l)}$ is also updated as follows:

$$\begin{aligned} b_{i,j}^{(l)} &= b_{i,j}^{(l)} + a_{i,j}^{(l)}, \\ a_{i,j}^{(l)} &= p_{j|i}^{(l)} \dot{q}_j^{(l+1)} \end{aligned} \tag{13}$$

where $a_{i,j}^{(l)}$ indicates the agreement between each vote and vote cluster. After R iterations, we obtain higher-level graph capsules $e^{(l+1)}$. Figure 2b shows that we add a residual connection to each pair of consecutive graph capsule layers to provide richer information for higher-level graph capsules.

To reduce the number of trainable parameters in KHGCN, we set $d_h = d$ in our experiments. It has been proven that such a reduction does not affect the final recommendation performance. Formally, the output of the graph capsule in layer $(l+1)$ is defined as $e^{(l+1)} \leftarrow e^{(l+1)} + GA(e^{(l)})$, where GA indicates the global average operation.

3.5. Model Prediction

After hierarchical graph capsules perform L-order information propagation, we obtain all representations of user node u at the L-level in the graph, namely $\{e_u^{(1)}, \ldots, e_u^{(L)}\}$. Likewise, we obtain the representation of item node v, namely$\{e_v^{(1)}, \ldots, e_v^{(L)}\}$. Therefore, we adopt the general layer-aggregation method to concatenate the representations of each layer into a vector to represent the target node via

$$e_u^{*} = [e_u^{(0)} || \ldots || e_u^{(L)}], e_v^{*} = [e_v^{(0)} || \ldots || e_v^{(L)}] \tag{14}$$

where $||$ is the concatenate mechanism for vectors.

We enrich the initial embedding by the embedding propagation operation and control the strength of information propagation by adjusting the parameter L simultaneously. In the prediction layer, we use an inner product operation on the vector of the user and item to predict the interaction probability between them via

$$\hat{y}(u,v) = e_u^{*T} e_v^{*} \tag{15}$$

3.6. Loss Function and Model Training

Training for TransR considers the relative order of positive and negative triples pairwise and trains with the pairwise ranking loss for overall ordering via

$$\mathcal{L}_{KG} = \sum_{h,r,t,t' \in \mathcal{F}} -ln\sigma(g(h,r,t') - g(h,r,t)) \tag{16}$$

where $\mathcal{F} = (h,r,t,t')|(h,r,t) \in \mathcal{G}, (h,r,t') \notin \mathcal{G}$, and (h,r,t') is the negative triple constructed by randomly replacing one entity in the positive triple, and $\sigma(\cdot)$ is the sigmoid function.

This layer models the entities and relationships through the graph triples. The knowledge graph loss function $\mathcal{L}_{KG}$ is used as a regularization item for the auxiliary recommendation to train together with the recommendation model, thereby improving the generalization ability of the recommendation model.

Likewise, our training construct for the recommendation part comes after obtaining the output $\hat{y}_{uv}$ of the prediction layer. In the prediction layer, the likelihood of user u and item v is estimated by feeding their final representations into the prediction function. We also use pairwise BPR loss to optimize the recommended partial model parameters $\Theta = \{u, v | u \in U, v \in V\}$. Specifically, since generated user–item interactions are identified user preferences, it is assumed that identified interactions should be assigned higher predictive values than unidentified interactions:

$$\mathcal{L}_{Rec} = \sum_{u,v,w \in \mathcal{O}} -ln\sigma(\hat{y}(u,v) - \hat{y}(u,w)) \tag{17}$$

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

Finally, we have the objective function to learn Equation (16) and Equation (17) jointly, as follows:

$$\mathcal{L}_{KHGCN} = \mathcal{L}_{KG} + \mathcal{L}_{Rec} + \lambda_\Theta \|\Theta\|_2^2 \tag{18}$$

where $\Theta = \{\mathrm{E}, \mathrm{W}_r, \forall l \in \mathcal{R}, \mathrm{W}_1^{(l)}, \mathrm{W}_2^{(l)}, \forall l \in \{1, \ldots, L\}\}$ is the model parameter set, E is the embedding representation of all entities and relations, and L2 regularization is performed on λ_Θ to prevent overfitting, where Θ is a parameter for controlling the regularization.

The whole training process is detailed in Algorithm 1. We optimize $\mathcal{L}_{KG}$ and $\mathcal{L}_{Rec}$ alternatively, and mini-batch Adam [57] is used to optimize the loss function $\mathcal{L}_{KHGCN}$ of the collaborative knowledge graph recommendation model. Adam can adaptively control the absolute value of the learning rate with respect to the gradient and is an ordinary universal optimizer in deep learning.

Algorithm 1 Procedure of KHGCN.

Input: users u; items v; Interaction matrix Y; knowledge graph $\mathcal{G}$

Output: recommend the top@N item list

```
1:  initialize all parameters, shuffle ($u$,$v$,$Y$,$\mathcal{G}$);
2:  for $i = 1$ to $|U| + |V| + |E|$ do
3:      apply TransR to obtain the embedding of the graph $\mathcal{G}$
4:      for $k = 1$ to $K$ do
5:          $z_{i,k} = \sigma(W_k^T \dot{x}_i) + b_k$
6:      end for
7:  end for
8:  obtain the $\mathcal{L}_{KGs}$ loss of the TransR from Equation (16);
9:  $u_i^{(1)} = squash(z_i)$
10: for $l = 1$ to $L$ do
11:     $b_{i,j}^{(l)} = 0$
12:     for $j = 1$ to $|N_i^{(l+1)}|$ do
13:         $p_j^{(l)} = TGAT_j(e_i^{(l)}, e_{N_i}^{(l)})$
14:     end for
15:     for $r = 1$ to $R$ do
16:         $c_{i,j}^{(l)} = exp(b_{i,j}^{(l)}) / \sum_s exp(b_{i,s}^{(l)})$
17:         $e_j^{(l+1)} = squash(\sum_i c_{i,j}^{(l)} p_{j|i}^{(l)})$
18:         $b_{i,j}^{(l)} = b_{i,j}^{(l)} + p_{j|i}^{(l)} \cdot e_j^{(l+1)}$
19:     end for
20:     $e_i^{(l+1)}$=GA($e_1^{(l+1)}$ | |,...,| | $e_j^{(l+1)}$ | |,...,| | $e_{|N_i^{(l+1)}|}^{(l+1)}$)
21:     $e_i^{(l+1)} = e_i^{(l+1)} + GA(e_i^{(l)})$
22: end for
```

Algorithm 1 *Cont.*

```
23: Concatenate $(e_u^{(1)}, \ldots, e_u^{(L)})$,Concatenate $(e_v^{(1)}, \ldots, e_v^{(L)})$
24: from prediction layer to calculate predicted probability $\hat{y}_{uv}$;
25: compute the cross-entropy loss of $\mathcal{L}_{Rec}$ from Equation (17);
26: compute the total-loss $\mathcal{L}_{KHGCN}$ from Equations (18);
27: Apply Equations (16)–(18) to obtain the back-propagated loss error and update parameters through the entire network;
28: update weights by the optimizer and update the learning rate $a$; return top@$N$ item list;
```

4. Experiment

In this section, to evaluate the effectiveness of our model, we carry out experiments on three real-world benchmark datasets, Amazon-Book, Last-FM, and Yelp2018, which are publicly accessible. Furthermore, the three datasets are different in application fields, and there are some differences in data size and terms of data sparsity as shown in Table 1.

Table 1. Details of the three datasets.

Datasets		Amazon-Book	Last-FM	Yelp2018
User–Item Interaction	#Users	70,679	23,566	45,919
	#Items	24,915	48,123	45,538
	#Interactions	847,733	3,034,796	1,185,068
	#Density	0.000481	0.002676	0.0005667
Knowledge Graph	#Entities	88,572	58,266	90,961
	#Relations	39	9	42
	#Triplets	2,557,746	464,567	1,853,704

4.1. Dataset Description

- **Amazon-Book**: Amazon-review is a widely used product recommendation dataset [58]. Amazon product data include some user–item data provided by Amazon. These data contain various Amazon products, such as books, electronics, movies and TV, home kitchens, outdoor sports, digital music, musical instruments, etc. This dataset includes reviews (ratings, text, and help votes), product metadata (description, category information, price, branding, and image features), and links (see/also buy charts). We choose Amazon-Book from Amazon for model performance evaluation, and we keep users and items with at least ten interactions (10-core).
- **Last-FM**: This is a dataset that provides music recommendations. Listening records of 92,800 singers from 1892 users; each user in the dataset contains a list of their most popular artists and the number of plays. It also includes user-applied labels that can be used to construct content vectors from the music listening dataset collected from the Last-FM online music system. Among them, tracks are considered items. In particular, we take a subset of the dataset with timestamps from January 2015 to June 2015. We utilize the same 10-core setup to ensure data quality.
- **Yelp2018**: This dataset was adopted from the 2018 edition of the Yelp challenge. This dataset covers business, review, and user data, and can be used for personal, educational, and academic purposes. Here, we consider local businesses such as restaurants and hotels as projects. Again, we use a 10-core setting to ensure at least ten interactions per user and item.
  To build a collaborative knowledge graph for recommendations, we introduce a knowledge graph based on the original user–item interactions in each dataset. As shown in Figure 1, the knowledge graph is introduced through the item side in our model. We first traverse each item in the Amazon book and LastFM datasets. If the mapping between the item and the knowledge graph is available (there is an intersection), we map the item to the Freebase entity through the title-matching method used in KGAT [19]. For Yelp2018, we extract item knowledge from local business

information networks as KG data. To guarantee the integrity of the constructed knowledge-aware dataset, we consider all triples directly related to item-aligned entities. The statistics of the three knowledge-aware datasets are shown in Table 1 When constructing knowledge graph data, to guarantee the quality of the constructed knowledge graph, we filter the uncommon entities (that is, fewer than 10 in the intersection of both datasets) and retain at least 50 relations that appear in triples to preprocess the knowledge graph part of the three datasets. We randomly select 80% of each user's interaction history for each dataset to form the training set to train the model parameters and the remaining 20% as the test set to verify the model performance. We randomly select 10% of each user's interactions from the training set as the validation set to tune the model's hyperparameters. We take each of its observed user–item interactions as a positive example and then pair it with a negative example with which the user has not interacted before through a negative sampling strategy.

4.2. Baseline Methods

- **FM** [2]: FMs model all interactions between the inputs variables using factorized parameters. Here, we take the ids of the constructed collaborative knowledge graph as input features for FM.
- **NFM** [59]: This method is a state-of-the-art decomposition model that uses the properties of neural networks to fit arbitrary functions to highlight generalized NFMs, treating FMs as a particular case of NFMs.
- **BPRMF** [60]: Most algorithms are based on user predictions of product ratings for implicit feedback data. From the perspective of sorting, BPRMF sorts according to each user's preference, in which the top-ranked items have higher priority.
- **CKE** [33]: Collaborative knowledge-base embedding (CKE) leverages the additional information in the knowledge base to improve the quality of recommendation systems and learns latent representations of entities related to items in collaborative filtering from the knowledge base.
- **ECFKG** [37]: The ECFKG model applies TransE [16] to the unified graph to embed relevant entities in the knowledge graph to enhance recommendations. Furthermore, a soft-matching KGE-based method is proposed to generate interpretable personalized recommendation lists for users.
- **KGAT** [19]: This model studies the utility of knowledge graphs, which breaks the independent interactions assumption by associating items with their attributes, and builds an end-to-end graph attention neural network approach to model higher-order connections in KG.
- **DGCF** [43]: Disentangled graph collaborative filtering (DGCF) models enhance user intent to disentangle these factors and yield disentangled representations. The specific method is to split the user's embedding into several segments, each segment representing a specific intent of a user.

4.3. Evaluation Metrics

To better examine that the proposed model works under real-world datasets, we evaluate the model by Precision@N, Recall@N, F1@N, and NDCG@N. $\hat{R}_N$ is a list of the top@N predicted for target user u, N is the length of the recommendation list from candidate items, and R is the test set.

Precision@N: Indicates the proportion of the predicted positive samples that are positive samples.

$$\text{Precision@N} = \frac{|R \bigcap \hat{R}_N|}{N} \tag{19}$$

Recall@N: Indicates that the predicted result is the proportion of the actual positive samples in the positive samples to the positive samples in the full sample.

$$\text{Recall@N} = \frac{|R \bigcap \hat{R}_N|}{|R|} \tag{20}$$

F1@N: The F1 score is a weighted average of precision and recall. The higher the F1 score, the more robust the model.

$$\text{F1@N} = 2 \cdot \frac{\text{Precision@N} \cdot \text{Recall@N}}{\text{Precision@N} + \text{Recall@N}} \tag{21}$$

NDCG (Normalized Discounted Cumulative Gain): The score of NDCG represents the correlation between the recommendation list and the target user.

$$\text{NDCG@N} = n_u \frac{\sum_j 2^{r(z)}}{log(1+z)} \tag{22}$$

In the above equation, n_u is a normalization constant. $r(z)$ is the correlation coefficient of the z-th item in the recommended list, which is generally set to an integer.

4.4. Experiment Settings

We implement our model in the Pytorch deep-learning framework. The embedding size is fixed to 64 for all models. All models are optimized by the optimizer Adam, which fixes the batch size to 4096. Model parameters are initialized using the default Xavier initializers [61]. We employ a generic grid search to determine model hyperparameters: the learning rate is adjusted in $\{0.05, 0.01, 0.005, 0.001\}$, L2 normalization coefficients are searched in $\{10^{-5}, 10^{-4}, \ldots, 10, 10^2\}$, and the dropout ratio is uniformly fixed to 0.1.

Furthermore, we employ an early-stopping mechanism during the validation process of model training, i.e., if the Recall@20 on the validation set does not increase in 100 consecutive epochs, the training is stopped early. To model the higher-order connectivity, we set the depth L of the model KHGCN to 3 and the hidden layer dimension (64, 32, 16). In Section 4.6.2, the effects of different layer depths L on the performance of the model KHGCN are analyzed experimentally. We adopt a bi-interaction strategy for each layer propagation in KHGCN to aggregate the input vectors.

To evaluate the model's performance for the target user, we take the 10 most recent interactions as the test set and use the rest of the data for training. The model performance is judged by the generated recommendation list and the four evaluation metrics: Precision@20, Recall@20, F1@20, and NDCG@20. We randomly divide each dataset into a training set, validation set, and test set, and average the results using 10-fold cross-validation for our model. Table 2 shows the hyperparameters set in different datasets, where d represents the latent dimension, L represents the depth of the KHGCN propagation, K represents the number of node-unwrapped latent subspaces, Rrepresents the number of routing iterations of the graph capsule, and λ_Θ represents the weight for L2 regularization.

For fairness, we set the parameters of other comparison algorithms to the same settings as those in Table 2, and other hyperparameters, except those in Table 2, are selected by grid search.

Table 2. Hyperparameters.

Datasets	Amazon-Book	Last-FM	Yelp2018
d	64	64	64
L	3	3	3
K	4	4	4
R	3	3	3
λ_Θ	0.0001	0.0001	0.00001

4.5. Performance Comparison

To verify the effectiveness of KHGCN for CKG-based recommendation tasks, we compare the results of KHGCN with other contrasting algorithms on three datasets and four evaluation metrics and show the result of the top@20 in Table 3.

- Three algorithms are first analyzed without considering the node embeddings: FM, NFM, and BRPMF. BPRMF outperforms FM and NFM because BPRMF benefits from sampling positive and negative samples (triples). Then, we train the model with BPR loss.
- Three algorithms are first analyzed without considering the node embeddings: neural network methods, such as CKE and CFKG, are significantly better than the above three methods, which verifies the effectiveness of the knowledge-graph embedding in the recommendation system. The results are shown in Table 3, which may reflect the contribution of deep learning methods to some extent.
- Table 3 shows that our model performs better and is more competitive than CKE and CFKG under the evaluation criteria. It verifies the superiority of the graph neural network structure in the case of graph-structure-based recommendations. To a certain extent, it reflects that our graph neural network model can provide target users with more accurate item recommendations.
- Compared with the representative graph-based recommendation method KGAT, Table 3 shows that KHGCN outperforms the KGAT method under four evaluation metrics of the three datasets. Specifically, KHGCN achieves a 5.4% average improvement. Because KGAT only builds a simple graph attention mechanism, it also shows the importance of utilizing the decoupling mechanism and the capsule graph network. We attribute the significant improvement to the expressiveness of graph neural networks in modeling multiple capsule propagation layers.
- DGCF achieves the best performance in the baseline, suggesting that the disentangled structure is beneficial for enhancing representation learning in the recommendation.

Finally, we analyze and summarize the experimental results of the proposed KHGCN. Our model, built with a decoupling mechanism and a capsule graph neural network, yields better performance than all contrasting algorithms. Experimental results validate the ability of KHGCN to model higher-order connections and learn user–item interactions in collaborative knowledge graphs. KHGCN consistently outperforms other comparison algorithms in all indicators in the performance comparison, which verifies the effectiveness of KHGCN enhanced representation learning to a certain extent.

Table 3. Performance of different methods for the top@20 recommendation.

	Amazon-Book				**Last-FM**				**Yelp2018**			
	Precision	**Recall**	**F1**	**NDCG**	**Precision**	**Recall**	**F1**	**NDCG**	**Precision**	**Recall**	**F1**	**NDCG**
FM	0.0144	0.1449	0.0262	0.0722	0.0306	0.0791	0.0442	0.0647	0.0157	0.0657	0.0253	0.0415
NFM	0.0139	0.1383	0.0252	0.0725	0.0293	0.0768	0.0424	0.0610	0.0144	0.0623	0.0235	0.0397
BPRMF	0.0142	0.1336	0.0256	0.0694	0.0316	0.0757	0.0446	0.0649	0.0163	0.0670	0.0262	0.0432
ECFKG	0.0130	0.1225	0.0236	0.0612	0.0293	0.0744	0.0420	0.0617	0.0147	0.0600	0.0236	0.0386
CKE	0.0145	0.1384	0.0263	0.0723	0.0315	0.0756	0.0445	0.0648	0.0165	0.0676	0.0265	0.0438
KGAT	0.0149	0.1416	0.0270	0.0756	0.0329	0.0802	0.0466	0.0688	0.0163	0.0670	0.0262	0.0429
DGCF	0.0151	0.1524	0.0277	0.0756	0.0320	0.0825	0.0463	0.0682	0.0164	0.0693	0.0263	0.0431
KHGCN	0.0170	0.1638	0.0308	0.0934	0.0374	0.0886	0.0526	0.0795	0.0184	0.0748	0.0294	0.0477

4.6. Analysis of Our Model

In this section, to further deepen the understanding of the KHGCN model, we analyze the more essential hyperparameters and components in the KHGCN model. Firstly, we experimentally verify the input dimension of the knowledge graph matching our model. Next, we investigate the sensitivity of KHGCN to the subspace number of disentanglement K and the iterations number of graph routing R. Third, we examine how changes in different hyperparameters during training affect the model performance. Moreover, we investigate how to select the aggregation mechanism that is most suitable for information dissemination in graph neural networks. Furthermore, we explore the effect of varying the number of layers L in graph neural networks on model performance. Lastly, we study how versions are affected by an additional component in our model.

4.6.1. Sensituvity Studies

In this section, we analyze the sensitivity of KHGCN to the subspace number of latent factors K = 2, 8 and the number of routing iterations R = 1, 2, 4, 5, where our method has the setting K = 4 and R = 3. As shown in Figure 3 and Table 4, the results show that KHGCN is not very sensitive to the two hyperparameters. Although the model's performance improves when K = 8 compared to when K = 4, the computational complexity of the unwrapping part also doubles. Similarly, although the evaluation index Recall@20 on the dataset Yelp2018 has slightly improved vehicle ability, the model requires more routing iterations when R = 4.

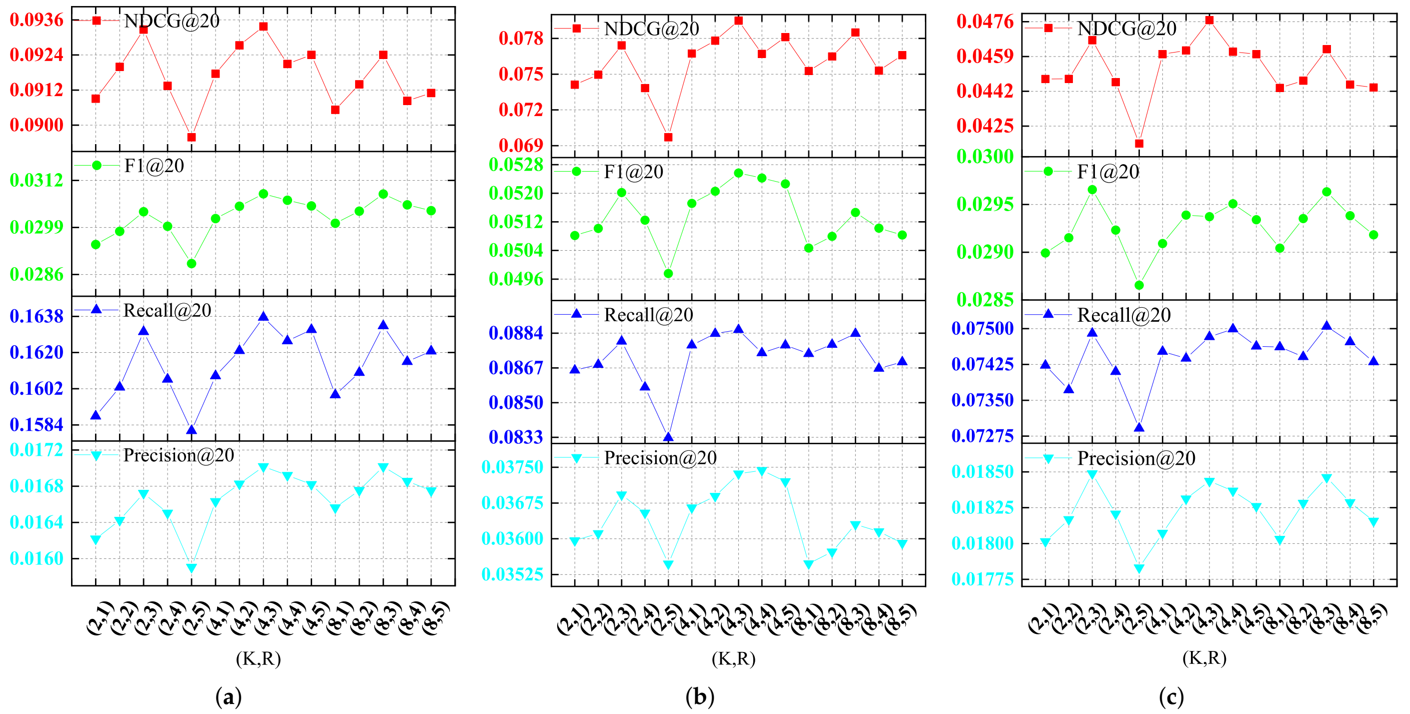

Figure 3. Sensitivity studies for the top@20 recommendation. (**a**) Amazon-Book. (**b**) Last-Fm. (**c**) Yelp2018.

Table 4. Sensitivity studies for the top@20 recommendation.

		Amazon-Book				Last-FM				Yelp2018			
		Precision	**Recall**	**F1**	**NDCG**	**Precision**	**Recall**	**F1**	**NDCG**	**Precision**	**Recall**	**F1**	**NDCG**
K	2	0.0167	0.1630	0.0303	0.0933	0.0370	0.0880	0.0520	0.0774	0.0185	0.0749	0.0297	0.0467
	8	0.0170	0.1633	0.0308	0.0924	0.0363	0.0884	0.0515	0.0785	0.0185	0.0750	0.0296	0.0463
R	1	0.0166	0.1608	0.0301	0.0918	0.0367	0.0878	0.0517	0.0767	0.0181	0.0745	0.0291	0.0460
	2	0.0168	0.1621	0.0304	0.0927	0.0369	0.0884	0.0521	0.0778	0.0183	0.0744	0.0294	0.0462
	4	0.0169	0.1626	0.0307	0.0921	0.0374	0.0874	0.0524	0.0767	0.0184	0.0750	0.0295	0.0461
	5	0.0168	0.1631	0.0305	0.0924	0.0372	0.0878	0.0523	0.0781	0.0183	0.0746	0.0293	0.0460
K = 4, R = 3		0.0170	0.1638	0.0308	0.0934	0.0374	0.0886	0.0526	0.0795	0.0184	0.0748	0.0294	0.0477

4.6.2. Effect of Embedding Propagation Layer Numbers

We test if the more propagation layers KHGCN is embedded with, the better the model performance.In particular, we set the embedding propagation layer L in [1, 2, 3, 4, 5].

Figure 4 and Table 5 show the performance of our model under different embedding propagation layers. Figure 4 straightforwardly shows that an appropriate increasing of the depth of the model can improve the performance of our model to a certain extent.

Notably, KHGCN-2 and KHGCN-3 significantly outperform KHGCN-1. Experiments show that KHGCN can effectively model higher-order relationships among the nodes in collaborative knowledge graphs, benefiting from the higher-order connectivity of the second- and third-order embedding layers relative to the first-order one. Further stacking too many layers in KHGCN, the performance drops, like with KHGCN-4 and KHGCN-5. The results show that KHGCN may be too profoundly degraded by the influence of noisy nodes embedded in the propagation layer. An excessively deep embedded propagation layer can lead to overfitting and reduce the model's performance.

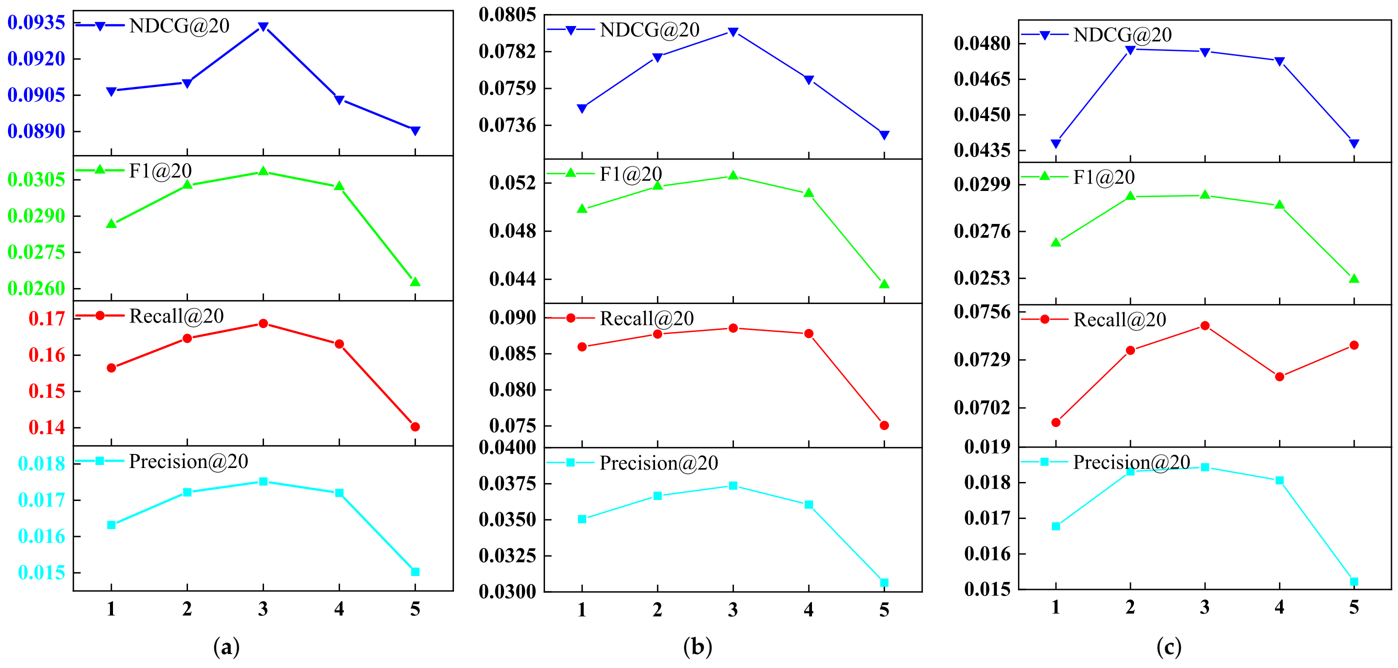

Figure 4. Effect of embedding propagation layer numbers for the top@20 recommendation. (**a**) Amazon-Book. (**b**) Last-Fm. (**c**) Yelp2018.

Table 5. Effect of embedding propagation layer numbers for the top@20 recommendation.

	Amazon-Book				**Last-FM**				**Yelp2018**			
	Precision	**Recall**	**F1**	**NDCG**	**Precision**	**Recall**	**F1**	**NDCG**	**Precision**	**Recall**	**F1**	**NDCG**
$L = 1$	0.0158	0.1515	0.0286	0.0907	0.0350	0.0860	0.0498	0.0747	0.0168	0.0694	0.0270	0.0438
$L = 2$	0.0167	0.1596	0.0303	0.0910	0.0367	0.0877	0.0517	0.0779	0.0183	0.0734	0.0293	0.0478
$L = 3$	0.0170	0.1638	0.03088	0.0934	0.0374	0.0886	0.0526	0.0795	0.0184	0.0748	0.0294	0.0477
$L = 4$	0.0167	0.1581	0.0302	0.0903	0.0361	0.0878	0.0511	0.0765	0.0181	0.0720	0.0289	0.0473
$L = 5$	0.0145	0.1352	0.0262	0.0891	0.0306	0.0751	0.0435	0.0731	0.0152	0.0737	0.0252	0.0438

4.6.3. Aggregators Analysis

To investigate how different aggregators for ego representations and neighbor representations will affect the performance of KHGCN, we perform experiments for four variants of KHGCN. Note that two single aggregators (GCN and GraphSage) and two Bi-interaction aggregators (Concatenate and Sum) are four variants of KHGCN. Table 6 show that Bi-interaction (Sum) outperforms GCN, GraphSage, and Bi-interaction (Concatenate) on both metrics. We attribute the improvement to feature interactions, which model affinities between the ego representation and neighbor representations. The result justifies the effectiveness and rationality of the Bi-interaction (Sum) aggregator to capture the heterogeneity of these four representations.

Table 6. Aggregators analysis for the top@20 recommendation.

Aggregators	**Amazon-Book**				**Last-FM**				**Yelp2018**			
	Precision	**Recall**	**F1**	**NDCG**	**Precision**	**Recall**	**F1**	**NDCG**	**Precision**	**Recall**	**F1**	**NDCG**
GCN	0.0165	0.1568	0.0298	0.0904	0.0361	0.0857	0.0508	0.07697	0.0177	0.0720	0.0285	0.0461
GraphSage	0.0168	0.1605	0.0304	0.0921	0.0359	0.0868	0.0508	0.07707	0.0178	0.0734	0.0286	0.0462
Concatenate	0.0167	0.1615	0.0303	0.0918	0.0367	0.0872	0.0517	0.0782	0.0181	0.0735	0.0291	0.0469
Sum	0.0170	0.1638	0.0308	0.0934	0.0374	0.0886	0.0526	0.0795	0.0184	0.0748	0.0294	0.0477

Comprehensive ablation studies are carried out in this section to understand the contribution of each component (i.e., disentangled graph capsules, attention layer, and capsule layers) in our method. We perform the ablation experiments on the component of the KHGCN to analyze their influence on the model performance. Here, we would like to examine how each factor contributes to the final performance. The results evaluated by evaluations are shown in Table 7 and Figures 5–7. To see this, we prepare five variants for comparison:

- w/o KGE: We disable the TransR embedding component of KHGCN. The variant removes the KG entities and their links from HKG but keeps the other nodes and links.
- w/o Att: We disable the attention mechanism and set $\pi(h,r,t)$ as $1/|N_h|$. The variant replaces the TGAT in the primary capsules with an average pool GNN component.
- w/o K&A: We obtain the variant by removing both components (w/o KGE and w/o Att).
- w/o Dis: Directly use the input node representation to serve as graph capsules, without considering the disentanglement factors. The variant removes the nodes disentangled from KHGCN but retains the KG entities and their links.
- w/o Res: remove the residual connection among the adjacent capsule layers.

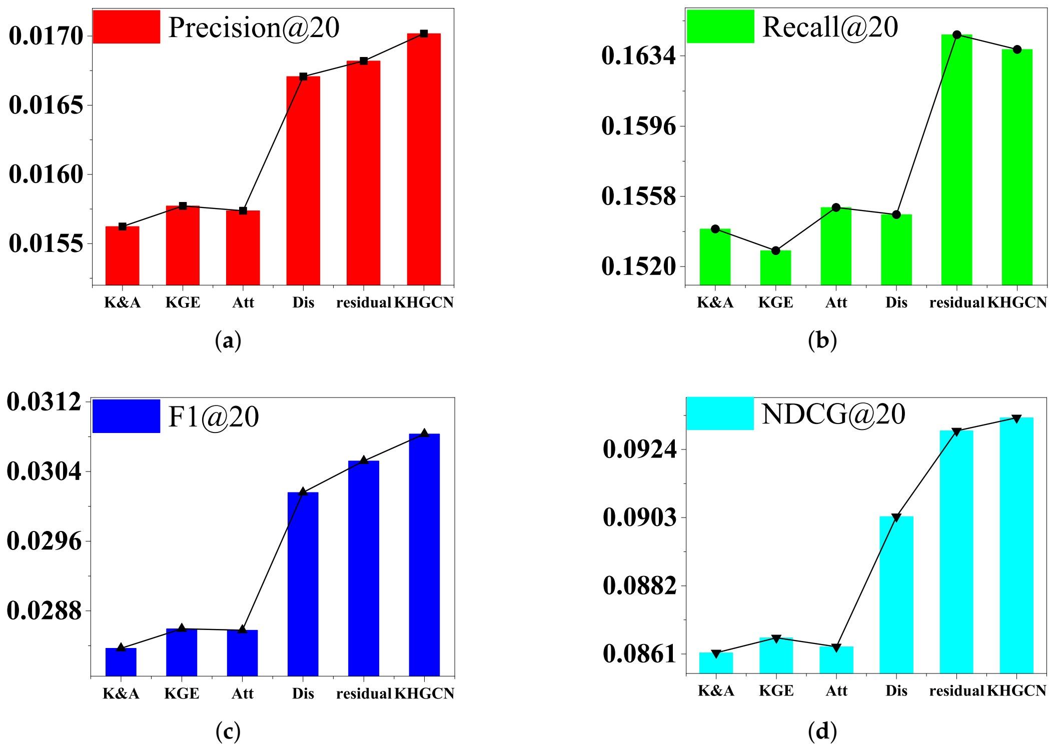

Figure 5. Ablation analysis on Amazon-Book for the top@20 recommendation. (**a**) Precision@20. (**b**) Recall@20. (**c**) F1@20. (**d**) NDCG@20.

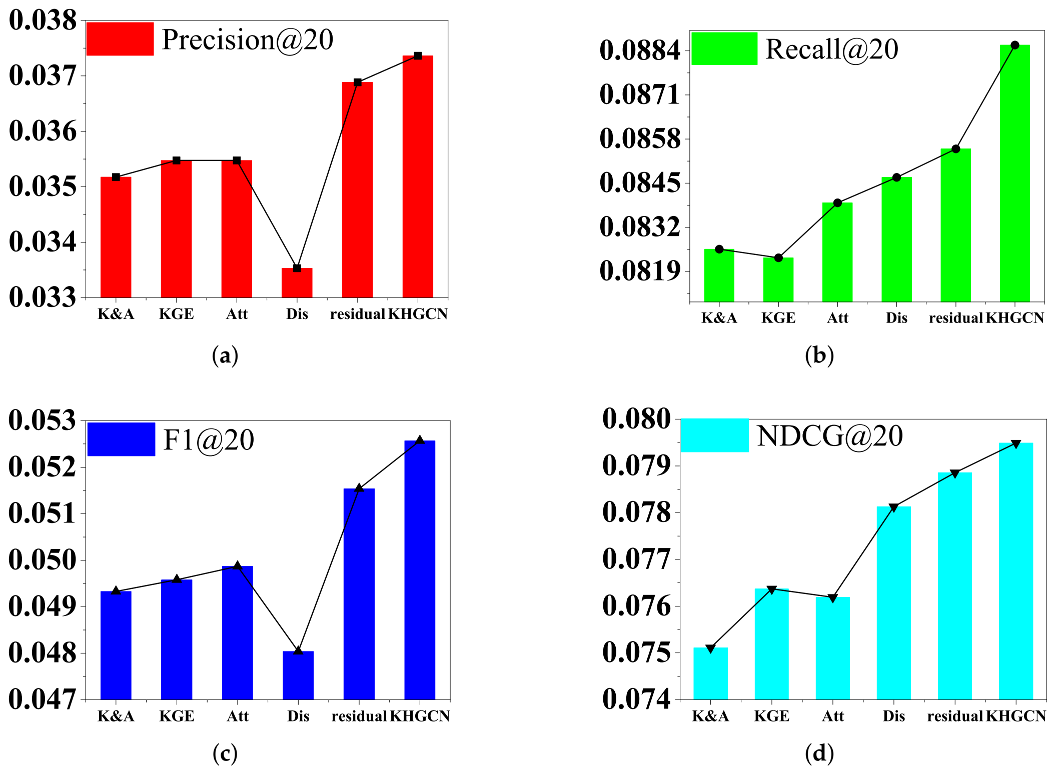

Figure 6. Ablation analysis on Last-Fm for the top@20 recommendation. (**a**) Precision@20. (**b**) Recall@20. (**c**) F1@20. (**d**) NDCG@20.

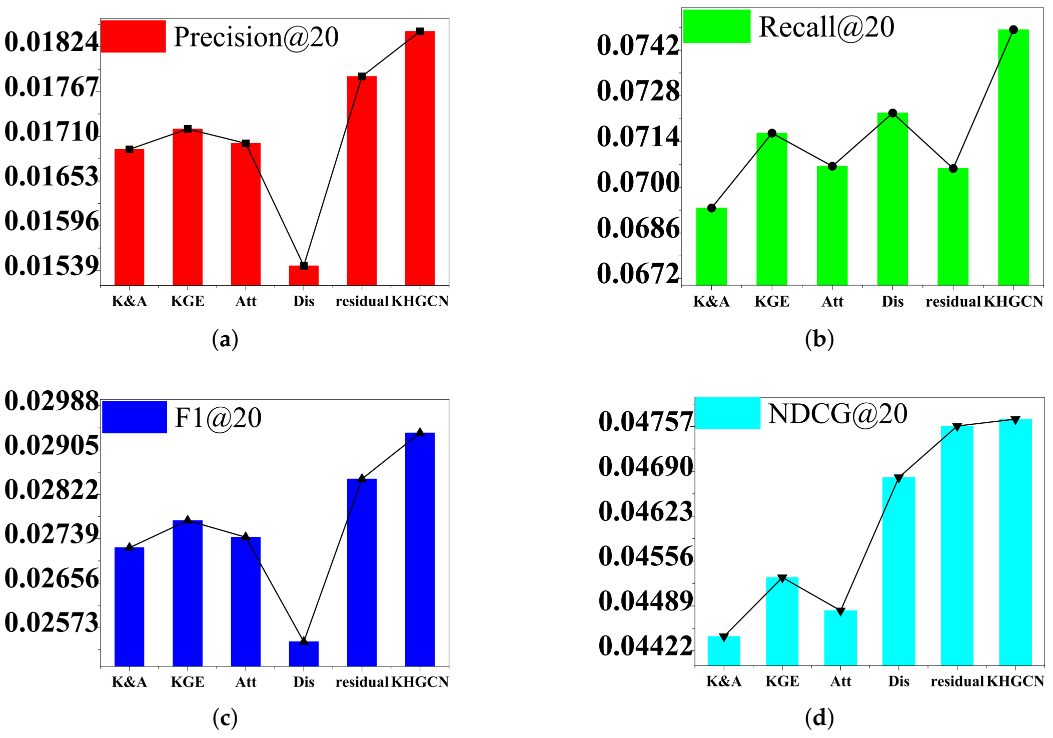

Figure 7. Ablation analysis on Yelp2018 for the top@20 recommendation. (**a**) Precision@20. (**b**) Recall@20. (**c**) F1@20. (**d**) NDCG@20.

Table 7. Ablation analysis for the top@20 recommendation.

	Amazon-Book				Last-FM				Yelp2018			
	Precision	**Recall**	**F1**	**NDCG**	**Precision**	**Recall**	**F1**	**NDCG**	**Precision**	**Recall**	**F1**	**NDCG**
w/o K&A	0.0156	0.1540	0.0283	0.0861	0.0352	0.0826	0.0493	0.0751	0.0169	0.0693	0.0272	0.0444
w/o KGE	0.0158	0.1529	0.0286	0.0866	0.0355	0.0823	0.0496	0.0764	0.0172	0.0717	0.0277	0.0453
w/o Att	0.0157	0.1552	0.0286	0.0863	0.0355	0.0840	0.0499	0.0762	0.0170	0.0706	0.0274	0.0448
w/o Dis	0.0167	0.1548	0.0302	0.0903	0.0335	0.0847	0.0480	0.0781	0.0155	0.0723	0.0255	0.0468
w/o Res	0.0168	0.1645	0.0305	0.0930	0.0369	0.0855	0.0515	0.0789	0.0179	0.0706	0.0285	0.0476
KHGCN	0.0170	0.1638	0.0308	0.0934	0.0374	0.0886	0.0526	0.0795	0.0184	0.0748	0.0294	0.0477

Table 7 and Figures 5–7 show the performance comparison between the complete model and the five variants. We summarize the experimental results in Table 7, and the results illustrated reveal the following:

- Removing knowledge-graph embedding and attention components degrade the model's performance. w/o K&A consistently underperforms compared to w/o KGE and w/o Att. It makes sense since w/o K&A fails to explicitly model the representation relatedness on the granularity of triplets. We can see that removing KG data significantly affects the performance of our model, which further verifies the usefulness of KG data.
- Compared with w/o Att, w/o KGE performs better in most cases. One possible reason is that treating all neighbors equally might introduce noises and mislead the embedding propagation process. It verifies the substantial influence of graph attention mechanisms.
- The variant w/o Dis removing nodes disentanglement gives a worse result than the complete model, which shows that disentangling node representation allows us to characterize the latent factors underlying each node and, in turn, more accurately preserve the node/graph properties and capture the part–whole relationship.
- Furthermore, the w/o variant Residual dropping the residual component is worse than the complete model, which indicates that combining fine, low-layer information with coarse, high-layer information gives us the ability to enhance the final graph-level representation.

Thus, we conclude that each component in our method is necessary and contributes to performance improvement.

5. Conclusions and Future Work

This work argues that there are complex and valuable relationships in the collaborative knowledge graph, and the knowledge-graph-based recommendation is essential. With this in mind, we propose a recommendation model with a knowledge-graph capsule network that employs a disentanglement mechanism to handle input node embeddings and a hierarchical capsule graph neural network layer to model higher-order connections and enhance representation learning for collaborative knowledge graphs. Extensive comparative experiments on three large-scale real-world datasets validate the rationality and effectiveness of KHGCN for modeling user and item representations in CKGs. Furthermore, an in-depth analysis of KHGCN demonstrates the usefulness and necessity of the individual components that make up the model KHGCN. This work focuses on the accuracy of recommendations, while ignoring other supplemental inaccuracy metrics of recommendations, such as diversity, novelty, coverage, etc. This work only explores information from entities, relationships, and users, while ignoring information other than user–item interactions and knowledge graphs, which may result in inaccurate recommendations. In addition, most of the existing knowledge-aware recommendations focus on the strong connection between entities, but the user's fine-grained preference for the item is not easy to capture. Next, we plan to introduce more fine-grained recommendations through the introduction of multimodal aware information.

In the future, we will continue to conduct research in the following directions based on the current research results:

1. Diversify input information and increase advanced representation learning models;
2. Furthermore, to build a more comprehensive knowledge graph, we will explore extracting entity-related information from other sources, such as text, images, etc.

Author Contributions: Methodology, F.C. and G.Y.; Software, F.C.; Investigation, F.C., G.L. and W.Z.; Writing—original draft, F.C. and G.Y.; Writing—review & editing, Y.D. and G.Y.; Visualization, F.C.; Supervision, G.Y. and Y.D.; Funding acquisition, G.Y. and Y.D. All authors have read and agreed to the published version of the manuscript.

Funding: Our research fund is supported by National Foreign Expert Project (DL2022180003L), Ministry of Industry and Information Technology—Special Project for High Quality Development of Manufacturing Industry (G064620007, KY10600200059).

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Data will be made available on request.

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

References

1. Sarwar, B.M.; Karypis, G.; Konstan, J.A.; Riedl, J. Item-based collaborative filtering recommendation algorithms. In Proceedings of the Tenth International World Wide Web Conference, Hong Kong, China, 1–5 May 2001; pp. 285–295.
2. Rendle, S. Factorization Machines. In Proceedings of the 2010 IEEE International Conference on Data Mining, Sydney, NSW, Australia, 13–17 December 2010; pp. 995–1000. [CrossRef]
3. Lakshmanna, K.; Kaluri, R.; Gundluru, N.; Alzamil, Z.S.; Rajput, D.S.; Khan, A.A.; Haq, M.A.; Alhussen, A. A Review on Deep Learning Techniques for IoT Data. *Electronics* **2022**, *11*, 1604. [CrossRef]
4. Kumar, V.; Lalotra, G.S.; Sasikala, P.; Rajput, D.S.; Kaluri, R.; Lakshmanna, K.; Shorfuzzaman, M.; Alsufyani, A.; Uddin, M. Addressing Binary Classification over Class Imbalanced Clinical Datasets Using Computationally Intelligent Techniques. *Healthcare* **2022**, *10*, 1293. [CrossRef]
5. He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; Chua, T. Neural Collaborative Filtering. In Proceedings of the 26th International Conference on World Wide Web, Perth, Australia, 3–7 April 2017; pp. 173–182.
6. He, X.; He, Z.; Song, J.; Liu, Z.; Jiang, Y.G.; Chua, T.S. NAIS: Neural Attentive Item Similarity Model for Recommendation. *IEEE Trans. Knowl. Data Eng.* **2018**, *30*, 2354–2366. [CrossRef]
7. Wang, Z.; Xia, H.; Chen, S.; Chun, G. Joint representation learning with ratings and reviews for recommendation. *Neurocomputing* **2021**, *425*, 181–190. [CrossRef]
8. Chen, B.; Ding, Y.; Xin, X.; Li, Y.; Wang, Y.; Wang, D. AIRec: Attentive intersection model for tag-aware recommendation. *Neurocomputing* **2021**, *421*, 105–114. [CrossRef]
9. Shi, L.; Wu, W.; Hu, W.; Zhou, J.; Chen, J.; Zheng, W.; He, L. DualGCN: An Aspect-Aware Dual Graph Convolutional Network for review-based recommender. *Knowl. Based Syst.* **2022**, *242*, 108359. [CrossRef]
10. Zou, D.; Wei, W.; Mao, X.L.; Wang, Z.J.; Qiu, M.; Zhu, F.; Cao, X. Multi-level Cross-view Contrastive Learning for Knowledge-aware Recommender System. In Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, 11–15 July 2022.
11. Zhang, Y.; Yuan, M.; Zhao, C.; Chen, M.; Liu, X. Aggregating knowledge-aware graph neural network and adaptive relational attention for recommendation. *Appl. Intell.* **2022**, *52*, 17941–17953. [CrossRef]
12. Dong, C.; Ju, X.; Ma, Y. HRS: Hybrid Recommendation System based on Attention Mechanism and Knowledge Graph Embedding. In Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, Melbourne, Australia, 14–17 December 2021.
13. Liu, Y.; Miyazaki, J.; Chang, Q. Knowledge graph enhanced multi-task learning between reviews and ratings for movie recommendation. In Proceedings of the ACM/SIGAPP Symposium on Applied Computing, Virtual Event, 25–29 April 2022.
14. Zou, D.; Wei, W.; Wang, Z.J.; Mao, X.; Zhu, F.; Fang, R.; Chen, D. Improving Knowledge-aware Recommendation with Multi-level Interactive Contrastive Learning. In Proceedings of the 31st ACM International Conference on Information & Knowledge Management, Atlanta, GA, USA, 17–21 October 2022.
15. Xiao, H.; Huang, M.; Hao, Y.; Zhu, X. TransA: An Adaptive Approach for Knowledge Graph Embedding. *arXiv* **2015**, arXiv:1509.05490.
16. Bordes, A.; Usunier, N.; García-Durán, A.; Weston, J.; Yakhnenko, O. Translating Embeddings for Modeling Multi-relational Data. In Proceedings of the 27th Annual Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 5–8 December 2013; pp. 2787–2795.

17. 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; pp. 2181–2187.
18. Wang, Z.; Zhang, J.; Feng, J.; Chen, Z. Knowledge Graph Embedding by Translating on Hyperplanes. In Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, Québec City, QC, Canada, 27–31 July 2014; pp. 1112–1119.
19. Wang, X.; He, X.; Cao, Y.; Liu, M.; Chua, T. KGAT: Knowledge Graph Attention Network for Recommendation. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, 4–8 August 2019; Teredesai, A., Kumar, V., Li, Y., Rosales, R., Terzi, E., Karypis, G., Eds.; ACM: New York, NY, USA, 2019; pp. 950–958. [CrossRef]
20. Kipf, T.N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, 24–26 April 2017; Conference Track Proceedings. OpenReview.net, 2017.
21. Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; Bengio, Y. Graph Attention Networks. *arXiv* **2017**, arXiv:1710.10903.
22. 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, KDD 2018, London, UK, 19–23 August 2018; Guo, Y., Farooq, F., Eds.; ACM: New York, NY, USA, 2018; pp. 974–983. [CrossRef]
23. 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, Virtual Event, China, 25–30 July 2020; Huang, J., Chang, Y., Cheng, X., Kamps, J., Murdock, V., Wen, J., Liu, Y., Eds.; ACM: New York, NY, USA, 2020; pp. 639–648. [CrossRef]
24. Yang, J.; Zhao, P.; Rong, Y.; Yan, C.; Li, C.; Ma, H.; Huang, J. Hierarchical Graph Capsule Network. In Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence (EAAI 2021), Virtual Event, 2–9 February 2021; AAAI Press: Washington, DC, USA, 2021; pp. 10603–10611.
25. Verma, S.; Zhang, Z.L. Graph Capsule Convolutional Neural Networks. *arXiv* **2018**, arXiv:1805.08090.
26. Mallea, M.D.G.; Meltzer, P.; Bentley, P.J. Capsule Neural Networks for Graph Classification using Explicit Tensorial Graph Representations. *arXiv* **2019**, arXiv:1902.08399.
27. Zhang, B.; Li, X.; Xu, X.; Leung, K.; Chen, Z.; Ye, Y. Knowledge Guided Capsule Attention Network for Aspect-Based Sentiment Analysis. *IEEE ACM Trans. Audio Speech Lang. Process.* **2020**, *28*, 2538–2551. [CrossRef]
28. Li, J.; Li, S.; Zhao, W.X.; He, G.; Wei, Z.; Yuan, N.J.; Wen, J. Knowledge-Enhanced Personalized Review Generation with Capsule Graph Neural Network. In Proceedings of the CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, 19–23 October 2020; d'Aquin, M., Dietze, S., Hauff, C., Curry, E., Cudré-Mauroux, P., Eds.; ACM: New York, NY, USA, 2020; pp. 735–744. [CrossRef]
29. Zhang, X.; Chen, L. Capsule Graph Neural Network. In Proceedings of the 7th International Conference on Learning Representations (ICLR 2019), New Orleans, LA, USA, 6–9 May 2019.
30. Xu, L.; Zhang, S.; Song, G.; Wang, J.; Wu, T.; Liu, G. Taxonomy-Enhanced Graph Neural Networks. In Proceedings of the 31st ACM International Conference on Information & Knowledge Management, Atlanta, GA, USA, 17–21 October 2022.
31. Sabour, S.; Frosst, N.; Hinton, G.E. Dynamic Routing Between Capsules. 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; Guyon, I., von Luxburg, U., Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R., Eds.; MIT Press: Cambridge, MA, USA, 2017; pp. 3856–3866.
32. Gu, P.; Han, Y.; Gao, W.; Xu, G.; Wu, J. Enhancing session-based social recommendation through item graph embedding and contextual friendship modeling. *Neurocomputing* **2021**, *419*, 190–202. [CrossRef]
33. Zhang, F.; Yuan, N.J.; Lian, D.; Xie, X.; Ma, W. 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; Krishnapuram, B., Shah, M., Smola, A.J., Aggarwal, C.C., Shen, D., Rastogi, R., Eds.; ACM: New York, NY, USA, 2016; pp. 353–362. [CrossRef]
34. Dong, B.; Zhu, Y.; Li, L.; Wu, X. Hybrid collaborative recommendation of co-embedded item attributes and graph features. *Neurocomputing* **2021**, *442*, 307–316. [CrossRef]
35. Yin, G.; Chen, F.; Dong, Y.; Li, G. Knowledge-aware recommendation model with dynamic co-attention and attribute regularize. *Appl. Intell.* **2021**, *52*, 3807–3824. [CrossRef]
36. Wang, H.; Zhang, F.; Zhao, M.; Li, W.; Xie, X.; Guo, M. Multi-Task Feature Learning for Knowledge Graph Enhanced Recommendation. In Proceedings of the World Wide Web Conference, San Francisco, CA, USA, 13–17 May 2019; pp. 2000–2010.
37. Ai, Q.; Azizi, V.; Chen, X.; Zhang, Y. Learning Heterogeneous Knowledge Base Embeddings for Explainable Recommendation. *Algorithms* **2018**, *11*, 137. [CrossRef]
38. Chen, L.; Wu, L.; Hong, R.; Zhang, K.; Wang, M. Revisiting Graph Based Collaborative Filtering: A Linear Residual Graph Convolutional Network Approach. In Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI 2020), New York, NY, USA, 7–12 February 2020; AAAI Press: Washington, DC, USA, 2020; pp. 27–34.

39. Tan, Q.; Liu, N.; Zhao, X.; Yang, H.; Zhou, J.; Hu, X. Learning to Hash with Graph Neural Networks for Recommender Systems. In Proceedings of the WWW '20: The Web Conference 2020, Taipei, Taiwan, 20–24 April 2020; Huang, Y., King, I., Liu, T., van Steen, M., Eds.; ACM/IW3C2: New York, NY, USA, 2020; pp. 1988–1998. [CrossRef]
40. Wu, L.; Yang, Y.; Zhang, K.; Hong, R.; Fu, Y.; Wang, M. Joint Item Recommendation and Attribute Inference: An Adaptive Graph Convolutional Network Approach. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2020, Virtual Event, China, 25–30 July 2020; Huang, J., Chang, Y., Cheng, X., Kamps, J., Murdock, V., Wen, J.; Liu, Y., Eds.; ACM: New York, NY, USA, 2020; pp. 679–688. [CrossRef]
41. Jin, B.; Gao, C.; He, X.; Jin, D.; Li, Y. Multi-Behavior Recommendation with Graph Convolutional Networks. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2020, Virtual Event, China, 25–30 July 2020; Huang, J., Chang, Y., Cheng, X., Kamps, J., Murdock, V., Wen, J., Liu, Y., Eds.; ACM: New York, NY, USA, 2020; pp. 659–668. [CrossRef]
42. Gao, X.; Feng, F.; Huang, H.; Mao, X.; Lan, T.; Chi, Z. Food recommendation with graph convolutional network. *Inf. Sci.* **2022**, *584*, 170–183. [CrossRef]
43. Liu, Z.; Meng, L.; Zhang, J.; Yu, P.S. Deoscillated Graph Collaborative Filtering. *arXiv* **2020**, arXiv:2011.02100.
44. Ge, H.; Li, Q.; Meng, S.; Hou, J. CPGCN: Collaborative Property-aware Graph Convolutional Networks for Service Recommendation. In Proceedings of the 2022 IEEE International Conference on Services Computing (SCC), Barcelona, Spain, 10–16 July 2022; pp. 10–19.
45. Wang, F.; Li, Y.; Zhang, Y.; Wei, D. KLGCN: Knowledge graph-aware Light Graph Convolutional Network for recommender systems. *Expert Syst. Appl.* **2022**, *195*, 116513. [CrossRef]
46. Zhu, P.; Cheng, D.; Luo, S.; Yang, F.; Luo, Y.; Qian, W.; Zhou, A. SI-News: Integrating social information for news recommendation with attention-based graph convolutional network. *Neurocomputing* **2022**, *494*, 33–42. [CrossRef]
47. Yuan, L.; Zhao, J.; Sun, T.; Jiang, X.; Yang, Z.; Wang, X.; Geng, Y. LncRNA-Disease Association Prediction Based on Graph Neural Networks and Inductive Matrix Completion. In Proceedings of the Intelligent Computing Theories and Application—16th International Conference (ICIC 2020), Bari, Italy, 2–5 October 2020; Huang, D., Jo, K., Eds.; Springer: Berlin/Heidelberg, Germany, 2020; Volume 12464, pp. 262–269. [CrossRef]
48. Sun, J.; Zhang, Y.; Guo, W.; Guo, H.; Tang, R.; He, X.; Ma, C.; Coates, M. Neighbor Interaction Aware Graph Convolution Networks for Recommendation. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2020, Virtual Event, China, 25–30 July 2020; Huang, J., Chang, Y., Cheng, X., Kamps, J., Murdock, V., Wen, J., Liu, Y., Eds.; ACM: New York, NY, USA, 2020; pp. 1289–1298. [CrossRef]
49. Xu, Y.; Zhang, Y.; Guo, W.; Guo, H.; Tang, R.; Coates, M. GraphSAIL: Graph Structure Aware Incremental Learning for Recommender Systems. In Proceedings of the CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, 19–23 October 2020; d'Aquin, M., Dietze, S., Hauff, C., Curry, E., Cudré-Mauroux, P., Eds.; ACM: New York, NY, USA, 2020; pp. 2861–2868. [CrossRef]
50. Wu, L.; Yang, Y.; Chen, L.; Lian, D.; Hong, R.; Wang, M. Learning to Transfer Graph Embeddings for Inductive Graph based Recommendation. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2020, Virtual Event, China, 25–30 July 2020; Huang, J., Chang, Y., Cheng, X., Kamps, J., Murdock, V., Wen, J., Liu, Y., Eds.; ACM: New York, NY, USA, 2020; pp. 1211–1220. [CrossRef]
51. Wang, X.; Wang, R.; Shi, C.; Song, G.; Li, Q. Multi-Component Graph Convolutional Collaborative Filtering. *Proc. AAAI Conf. Artif. Intell.* **2020**, *34*, 6267–6274. [CrossRef]
52. Wu, L.; Li, J.; Sun, P.; Hong, R.; Ge, Y.; Wang, M. DiffNet++: A Neural Influence and Interest Diffusion Network for Social Recommendation. *IEEE Trans. Knowl. Data Eng.* **2020**, *34*, 4753–4766. [CrossRef]
53. Jiang, Y.; Ma, H.; Liu, Y.; Li, Z.; Chang, L. Enhancing social recommendation via two-level graph attentional networks. *Neurocomputing* **2021**, *449*, 71–84. [CrossRef]
54. Sun, R.; Cao, X.; Zhao, Y.; Wan, J.; Zhou, K.; Zhang, F.; Wang, Z.; Zheng, K. Multi-modal Knowledge Graphs for Recommender Systems. In Proceedings of the CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, 19–23 October 2020; d'Aquin, M., Dietze, S., Hauff, C., Curry, E., Cudré-Mauroux, P., Eds.; ACM: New York, NY, USA, 2020; pp. 1405–1414. [CrossRef]
55. Feng, Y.; Hu, B.; Lv, F.; Liu, Q.; Zhang, Z.; Ou, W. ATBRG: Adaptive Target-Behavior Relational Graph Network for Effective Recommendation. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2020, Virtual Event, China, 25–30 July 2020; Huang, J., Chang, Y., Cheng, X., Kamps, J., Murdock, V., Wen, J., Liu, Y., Eds.; ACM: New York, NY, USA, 2020; pp. 2231–2240. [CrossRef]
56. Hinton, G.E.; Sabour, S.; Frosst, N. Matrix capsules with EM routing. In Proceedings of the 6th International Conference on Learning Representations (ICLR 2018), Vancouver, BC, Canada, 30 April–3 May 2018.
57. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the 3rd International Conference on Learning Representations, San Diego, CA, USA, 7–9 May 2015.
58. He, R.; McAuley, J.J. Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering. In Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, QC, Canada, 11–15 April 2016; pp. 507–517. [CrossRef]

59. He, X.; Chua, T. 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; Kando, N., Sakai, T., Joho, H., Li, H., de Vries, A.P., White, R.W., Eds.; ACM: New York, NY, USA, 2017; pp. 355–364. [CrossRef]
60. Steffen, R; Christoph, F. BPR: Bayesian Personalized Ranking from Implicit Feedback. UAI 2009. In Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, 18–21 June 2009; Steffen, R., Christoph, F., Zeno, G., Lars, S., Eds.; AUAI Press: Quebec City, QC, Canada, 2009; pp. 452–461. Available online: https://dblp.org/rec/conf/uai/RendleFGS09.bib (accessed on 26 March 2023).
61. Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Sardinia, Italy, 13–15 May 2010; Teh, Y.W., Titterington, D.M., Eds.; JMLR, Inc.: Norfolk, MA, USA, 2010; Volume 9, pp. 249–256.