applied sciences

MDPI

Article

CKGAT: Collaborative Knowledge-Aware Graph Attention Network for Top-N Recommendation

Zhuoming Xu *, Hanlin Liu, Jian Li, Qianqian Zhang and Yan Tang

College of Computer and Information, Hohai University, Nanjing 210098, China; hanlin.liu@hhu.edu.cn (H.L.); jli@hhu.edu.cn (J.L.); qianqian.zhang@hhu.edu.cn (Q.Z.); tangyan@hhu.edu.cn (Y.T.)
* Correspondence: zmxu@hhu.edu.cn; Tel.: +86-25-58099120

Abstract: Knowledge graph-based recommendation methods are a hot research topic in the field of recommender systems in recent years. As a mainstream knowledge graph-based recommendation method, the propagation-based recommendation method captures users' potential interests in items by integrating the representations of entities and relations in the knowledge graph and the high-order connection patterns between entities to provide personalized recommendations. For example, the collaborative knowledge-aware attentive network (CKAN) is a typical state-of-the-art propagation-based recommendation method that combines user-item interactions and knowledge associations in the knowledge graph, and performs heterogeneous propagation in the knowledge graph to generate multi-hop ripple sets, thereby capturing users' potential interests. However, existing propagation-based recommendation methods, including CKAN, usually ignore the complex relations between entities in the multi-hop ripple sets and do not distinguish the importance of different ripple sets, resulting in inaccurate user potential interests being captured. Therefore, this paper proposes a top-N recommendation method named collaborative knowledge-aware graph attention network (CKGAT). Based on the heterogeneous propagation strategy, CKGAT uses the knowledge-aware graph attention network to extract the topological proximity structures of entities in the multi-hop ripple sets and then learn high-order entity representations, thereby generating refined ripple set embeddings. CKGAT further uses an attention aggregator to perform weighted aggregation on the ripple set embeddings, the user/item initial entity set embeddings, and the original representations of items to generate accurate user embeddings and item embeddings for the top-N recommendations. Experimental results show that CKGAT, overall, outperforms three baseline methods and six state-of-the-art propagation-based recommendation methods in terms of recommendation accuracy, and outperforms four representative propagation-based recommendation methods in terms of recommendation diversity.

Keywords: knowledge graph-based recommendation; top-N recommendation; user preference; heterogeneous propagation; graph attention network; attention aggregator

Citation: Xu, Z.; Liu, H.; Li, J.; Zhang, Q.; Tang, Y. CKGAT: Collaborative Knowledge-Aware Graph Attention Network for Top-N Recommendation. *Appl. Sci.* **2022**, *12*, 1669. https://doi.org/10.3390/app12031669

Academic Editor: Douglas O'Shaughnessy

Received: 30 December 2021
Accepted: 29 January 2022
Published: 5 February 2022

1. Introduction

Although recommendation technology has achieved considerable development and has been widely used in various domains, recommender systems still suffer from several challenges, such as inaccurate recommendations, data sparsity, and cold-start problems. In recent years, introducing a knowledge graph (KG) into the recommender system as side information has attracted considerable research interest, and the knowledge graph-based recommendation method has recently become a hot research topic in the field of recommender systems [1].

Knowledge graphs have proven to be effective in improving recommendation performance and alleviating the aforementioned challenges, because the knowledge graph can provide background knowledge for users and items in the recommender system, which helps to more accurately capture user preferences for items. According to how the

knowledge in the knowledge graph is utilized, knowledge graph-based recommendation methods can be classified into three categories [1]: embedding-based recommendation methods, connection-based recommendation methods, and propagation-based recommendation methods. The core idea of the embedding-based recommendation methods, such as collaborative knowledge-base embedding (CKE) [2], is leveraging the fruitful facts in the knowledge graph to enrich the representations of users and item. The core idea of connection-based recommendation methods, such as knowledge-aware path recurrent network (KPRN) [3], is using the connection patterns between entities (users/items) in the knowledge graph to guide recommendation. The core idea of the propagation-based recommendation methods, such as collaborative knowledge-aware attentive network (CKAN) [4], knowledge graph convolutional networks (KGCN) [5], and knowledge graph-based intent network (KGIN) [6], is integrating entity representations, relation representations, and high-order connection patterns between entities to provide more personalized recommendation. For a more comprehensive review of knowledge graph-based recommender systems, we refer readers to [1].

As commented in [1], the embedding-based recommendation methods are difficult to capture the high-order relations between entities, whereas the connection-based recommendation methods tend to lose information by decomposing the sophisticated connection pattern into separate linear paths. The propagation-based recommendation methods aim to make full use of the information in the knowledge graph, and thus become the mainstream methods for knowledge graph-based recommender systems. These methods adopt the idea of embedding propagation [7,8], and usually employ an architecture based on graph neural network [9] to refine entity representations by aggregating the embeddings of multi-hop neighbors in the knowledge graph to better predict user preferences for items. For example, CKAN, a typical state-of-the-art propagation-based recommendation method, combines user-item interactions and knowledge associations in the knowledge graph, and performs heterogeneous propagation in the knowledge graph to generate multi-hop ripple sets, thereby capturing users' potential interests. However, existing propagation-based recommendation methods, including CKAN, usually ignore the complex relations between entities in the multi-hop ripple sets [5,10] and do not distinguish the importance of different ripple sets, resulting in inaccurate user potential interests being captured.

In order to capture more accurate user potential interests in items, which is the key to improving the performance of personalized recommendation, this paper proposes a top-N recommendation method named collaborative knowledge-aware graph attention network (CKGAT), which consists of three layers: heterogeneous propagation, knowledge-aware graph attention network (GAT)-based attentive embedding, and user-item interaction probability prediction. Intuitively, different from existing propagation-based recommendation methods, CKGAT can distinguish the importance of different multi-hop ripple sets of either users or items and take full advantage of the topological proximity structures of entities in the multi-hop ripple sets to refine user representations and item representations, thus capturing more accurate potential user interests in items. Specifically, based on the heterogeneous propagation strategy, CKGAT employs the knowledge-aware GAT to extract the topological proximity structures of entities in the multi-hop ripple sets and then learn high-order entity representations, thereby generating refined ripple set embeddings. Furthermore, CKGAT adopts the attention aggregator to perform weighted aggregation on the ripple set embeddings, user/item initial entity set embeddings, as well as the original representations of items, so as to generate accurate user embeddings and item embeddings for top-N recommendations. We used four real-world datasets, including Last.FM, Book-Crossing, MovieLens 20M, and Dianping-Food, to empirically evaluate the performance of CKGAT. Experimental results show that CKGAT overall outperforms three baseline methods and six state-of-the-art propagation-based recommendation methods in terms of recommendation accuracy, and outperforms four representative propagation-based recommendation methods in terms of recommendation diversity.

The contributions of this work are summarized as follows:

1. We propose a novel method called collaborative knowledge-aware graph attention network (CKGAT) for top-N recommendation. This method can learn refined ripple set embeddings, thereby generating accurate user embeddings and item embeddings, so as to accurately capture users' potential interests in items. To the best of our knowledge, it is the first method that uses the knowledge-aware graph attention network to learn the refined ripple set embeddings;
2. We use the attention aggregator to generate accurate user embeddings and item embeddings for top-N recommendation;
3. Extensive experiments on four real-world datasets demonstrate the superiority of our CKGAT in comparison with state-of-the-art methods.

The remainder of this paper is organized as follows. In Section 2, we briefly summarize the related work focusing on the knowledge graph-based recommendation methods. After the problem formulation in Section 3, we describe in detail our proposed CKGAT method in Section 4. Section 5 presents the experiments. Finally, we conclude our work in Section 6.

2. Related Work

In this section, we briefly review the related work on knowledge graph-based recommendation methods, which are divided into three categories including embedding-based recommendation methods, connection-based recommendation methods, and propagation-based recommendation methods. The propagation-based recommendation methods are further divided into three types of approaches: user representation-refinement approaches, item representation-refinement approaches, as well as both user and item representation-refinement approaches.

2.1. Embedding-Based Recommendation Methods

The embedding-based recommendation methods use fruitful facts in the knowledge graph to enrich user representations and item representations [1]. This category of methods has two basic modules: the graph embedding module and the recommendation module [1]. The former is used to learn the embeddings of entities and relations encoded in knowledge graphs, and the latter is used to calculate user preferences for items. The embedding-based recommendation methods are relatively early knowledge graph-based recommendation methods, and have been widely used in the fields of word embedding, information retrieval, and recommendation [11]. For example, Zhang et al. [2] proposed the collaborative knowledge-base embedding (CKE) framework, which uses collaborative filtering and knowledge graph embedding to extract item features and capture the implicit relations between users and items. Wang H. et al. [12] proposed the deep knowledge-aware network (DKN), which uses a convolutional neural network (CNN) [13] and the knowledge graph embedding model TransD [14] to learn news embeddings for recommendations. Cao et al. [15] proposed the knowledge-enhanced translation-based user preference (KTUP) model, which can train both the knowledge graph completion task and the recommendation task at the same time to jointly learn user representations, item representations, entity representations, and relation representations. The embedding-based recommendation methods are simple to implement, but have difficulty capturing the high-order relations between entities, which causes the learned entity embeddings to be inaccurate, thus limiting recommendation performance.

2.2. Connection-Based Recommendation Methods

The connection-based recommendation methods use the connection patterns between entities (users/items) in a knowledge graph to guide recommendations. Traditional connection-based methods use meta-structures (such as meta-paths [16,17] and meta-graphs [18], etc.) to calculate the similarity between entities, and use it as a constraint for learning user and item representations [1]. For example, Luo et al. [19] used heterogeneous social networks for social recommendation and proposed a collaborative filtering algorithm, Hete-CF, based on metapath similarity. Some recent connection-based recommendation

methods provide recommendations by explicitly learning the embeddings of connection patterns [1,4,20]. For example, Sun et al. [21] proposed a recommendation method using recurrent knowledge graph embedding (RKGE), which uses a recurrent network [22] architecture to learn the embeddings of semantic paths between entities, thereby learning entity embeddings and path representations. Wang X. et al. [3] proposed the knowledge-aware path recurrent network (KPRN), which generates path representations by combining the semantics of entities and relations. Furthermore, KPRN employs a weighted aggregation operation to distinguish the importance of different paths, so as to capture the user preferences. The connection-based method decomposes the complex connection patterns in the knowledge graph into separate linear paths to learn path embeddings, which will inevitably lose information and cause the learned path embedding to be inaccurate. This will make the captured user preferences not accurate enough to produce accurate recommendation. Additionally, the connection-based recommendation method enumerates all possible paths between user-item pairs or item-item pairs. Therefore, when the scale of the knowledge graph is large and contains a large number of paths, this will inevitably reduce the scalability of the method. At the same time, this category of methods requires a high cost to calculate recommendations [1]. This may be the reason why the connection-based recommendation methods did not become the mainstream knowledge graph-based recommendation methods.

2.3. Propagation-Based Recommendation Methods

The embedding-based recommendation methods are simple to implement, but have difficulty capturing the high-order relations between entities. The connection-based recommendation methods use the connection patterns between entities in a knowledge graph to guide the recommendation, but they require high calculation cost and have poor scalability. In order to make full use of the information in the knowledge graph, scalable propagation-based recommendation methods capable of capturing the high-order relations between entities have recently been proposed and become the mainstream methods for knowledge graph-based recommender systems. The propagation-based recommendation method integrates entity representations, relation representations, and high-order connection patterns between entities to provide more personalized recommendation. According to the types of entities that are refined during the propagation process, the propagation-based recommendation methods can be further divided into three types of approaches [1]: user representation-refinement approaches, item representation-refinement approaches, and both user and item representation-refinement approaches.

2.3.1. User Representation-Refinement Approaches

The user representation-refinement approaches use the embeddings of the items interacted with by the user and the embeddings of the entities (i.e., items) in multi-hop ripple sets to learn user representations. For example, Wang H. et al. [23] proposed the RippleNet framework, which uses preference propagation to learn ripple set embeddings to generate user embeddings. Wang Z. et al. [4] proposed the collaborative knowledge-aware attentive network (CKAN) method, which uses a heterogeneous propagation strategy to obtain multi-hop ripple sets, and then inputs the ripple sets into the attentive network to learn user representations and item representations. He M. et al. [24] proposed the relation-enhanced knowledge graph reasoning for recommendation (RE-KGR) method, which calculates all user-item interaction paths to refine user embeddings, thereby generating recommendations. However, this type of methods has two limitations: one is that it ignores the topological proximity structures of the entities in multi-hop ripple sets, leading to inaccurate ripple set embeddings; the other is that it simply uses vector addition or concatenation operations to aggregate all the ripple set embeddings, thus reducing the accuracy of user/item embeddings.

2.3.2. Item Representation-Refinement Approaches

The item representation-refinement approaches refine item representations by aggregating the embeddings of items' multi-hop neighbors, and usually adopt graph neural network architectures suitable for the embedding propagation process [25,26]. For example, Wang H. et al. [5] proposed the knowledge graph convolutional networks (KGCN) framework, which uses a graph convolutional network [26,27] to iteratively aggregate the embeddings of neighboring entities to refine item representations. The authors [7] further proposed the knowledge-aware graph neural networks with label smoothness regularization (KGNN-LS) framework, in which a label-smoothness mechanism is added to the KGCN framework to propagate user-interaction labels, in order to simultaneously capture the semantic relations between entities as well as the users' personalized preferences for items, so as to provide effective recommendations. The gated knowledge graph neural networks for top-N recommendation system (GKGNN) method, proposed by Mu et al. [28], refines item representations by selecting more relevant neighboring entities in the process of aggregating neighboring item embeddings. The knowledge-aware conditional attention networks (KCAN) method, proposed by Tu et al. [29], condenses and refines the knowledge graph according to a given user, thus retaining useful information and refining item representations. The differentiable sampling on knowledge graph for recommendation with relational GNN (DSKReG) method proposed by Wang Y. et al. [30] adopts a differentiable sampling strategy, and uses the item categories and the relations between item entities to obtain the items related to the recommendations, thereby improving the item representations. However, this type of method only exploits the knowledge in knowledge graphs to learn item embeddings without leveraging the user-item interaction information.

2.3.3. Both User and Item Representation-Refinement Approaches

Different from the aforementioned two types of propagation-based recommendation methods, the both user and item representation-refinement approaches construct user–item knowledge graphs [1,8], a.k.a. collaborative knowledge graphs (CKGs), which contain users, items, item attribute entities, and the relations between them, and use graph neural networks to refine user embeddings and item embeddings. For example, Wang X. et al. [8] proposed the knowledge graph attention network (KGAT) method, which first uses TransR [31] to obtain the initial representations of the entities in a CKG, and then uses the graph attention network to refine user embeddings and item embeddings. Zhao et al. [32] proposed an extensible recommendation framework IntentGC, which uses graph convolutional networks to capture user preferences and heterogeneous relations in the knowledge graph, and then learns user representations and item representations for recommendations. Huang et al. [33] proposed the entity-aware collaborative relation network (ECRN) method, which uses a knowledge graph to explore the entity-granularity relation semantics behind user-item interactions, thus refining user representations and item representations. The X-2ch model proposed by Lo et al. [34] employs a four-channel mechanism to propagate information in the knowledge graph to obtain representative user representations and item representations. The knowledge graph-based intent network (KGIN) method, proposed by Wang X. et al. [6], uses auxiliary item knowledge to explore the users' intention behind the user-item interactions, and uses an information aggregation mechanism to refine the information related to the users' intention, and finally encodes this information in the user representations and the item representations to improve these representations. However, the CKGs that these methods rely on do not distinguish between user nodes and other entity nodes. Therefore, it is impossible to make full use of the collaborative signals in the user-item interactions to refine user representations and item representations, and it is difficult to deal with new users [4]. Our CKGAT method belongs to the both user and item representations refinement approaches. Different from the existing work, CKGAT can make full use of both the topological proximity structures of entities and the collaborative signals in user-item interactions to refine user representations and item representations, thus achieving better recommendation results.

3. Problem Formulation

This section first introduces some necessary concepts and mathematical symbols, and then describes the task of the propagation-based recommendation method.

Definition 1. *Recommender system. A recommender system involves the set $\mathcal{U}$ of m users and the set $\mathcal{V}$ of n items. For the system, a user-item interaction matrix, $\mathbf{Y} \in \mathbb{R}^{m \times n}$, is constructed according to the users' implicit feedback, where an element $y_{uv} = 1$ represents that user $u \in \mathcal{U}$ has interacted with item $v \in \mathcal{V}$ (e.g., clicking, watching, or purchasing), which is generally considered to indicate that the user likes the item; otherwise $y_{uv} = 0$.*

Definition 2. *Knowledge graph $\mathcal{G}$. A knowledge graph is defined as a set of triples, that is, $\mathcal{G} = \{(h, r, t) \mid h, t \in \mathcal{E}, r \in \mathcal{R}\}$, where $\mathcal{E}$ is the set of entities, $\mathcal{R}$ is the set of relations, and each triple (h, r, t) describes the relation $r \in \mathcal{R}$ from the head entity $h \in \mathcal{E}$ to the tail entity $t \in \mathcal{E}$. Note that $\mathcal{R}$ contains relations in both canonical direction (e.g., DirectorOf) and inverse direction (e.g., DirectedBy). In the graph, an entity is represented as a node, and a relation is represented as a directed edge (link) from the head entity node to the tail entity node.*

This paper uses a set $\mathcal{A} = \{(v, e) \mid v \in \mathcal{V}$ can be aligned with $e \in \mathcal{E}\}$ to describe the alignments between items in the recommender system and entities in the knowledge graph. An element $(v, e) \in \mathcal{A}$ indicates that item v can be aligned with entity e. The task of the propagation-based recommendation method is formulated in Definition 3.

Definition 3. *The task of the propagation-based recommendation method. As described in [4,5], given the user-item interaction matrix $\mathbf{Y}$ and the knowledge graph $\mathcal{G}$, the recommendation task is to predict the probability that user $u \in \mathcal{U}$ will interact with item $v \in \mathcal{V}$ that the user has not interacted with before. More specifically, the goal of the task is to learn a prediction function $\hat{y}_{uv} = \mathcal{F}(u, v \mid \Theta, \mathbf{Y}, \mathcal{G})$, where $\hat{y}_{uv}$ represents the predicted probability that user u will interact with item v, and Θ denotes the model parameters of function $\mathcal{F}$. Given a target user, the recommendation method sorts the predicted probabilities in descending order to generate a top-N recommendation list for the target user.*

4. Proposed CKGAT Method

This section elaborates on our proposed CKGAT method. First, we briefly describe the overall framework of CKGAT. Then, we explain in detail the three layers within the framework. Finally, we describe model learning and recommendation generation.

4.1. Overall Framework of CKGAT

The core ideas of CKGAT are as follows. On the basis of the heterogeneous propagation strategy, CKGAT uses the knowledge-aware graph attention network to extract the topological proximity structures of entities in the multi-hop ripple sets and then learn the high-order entity representations, thereby generating refined ripple set embeddings. CKGAT further uses the attention aggregator to perform weighted aggregation on the ripple set embeddings, the user/item initial entity set embeddings, and the original representations of items, so as to generate accurate user embeddings and item embeddings.

Figure 1 shows the overall framework of CKGAT, which consists of three layers: the heterogeneous propagation layer, the knowledge-aware GAT-based attentive embedding layer, and the user-item interaction probability prediction layer. These layers are briefly described as follows.

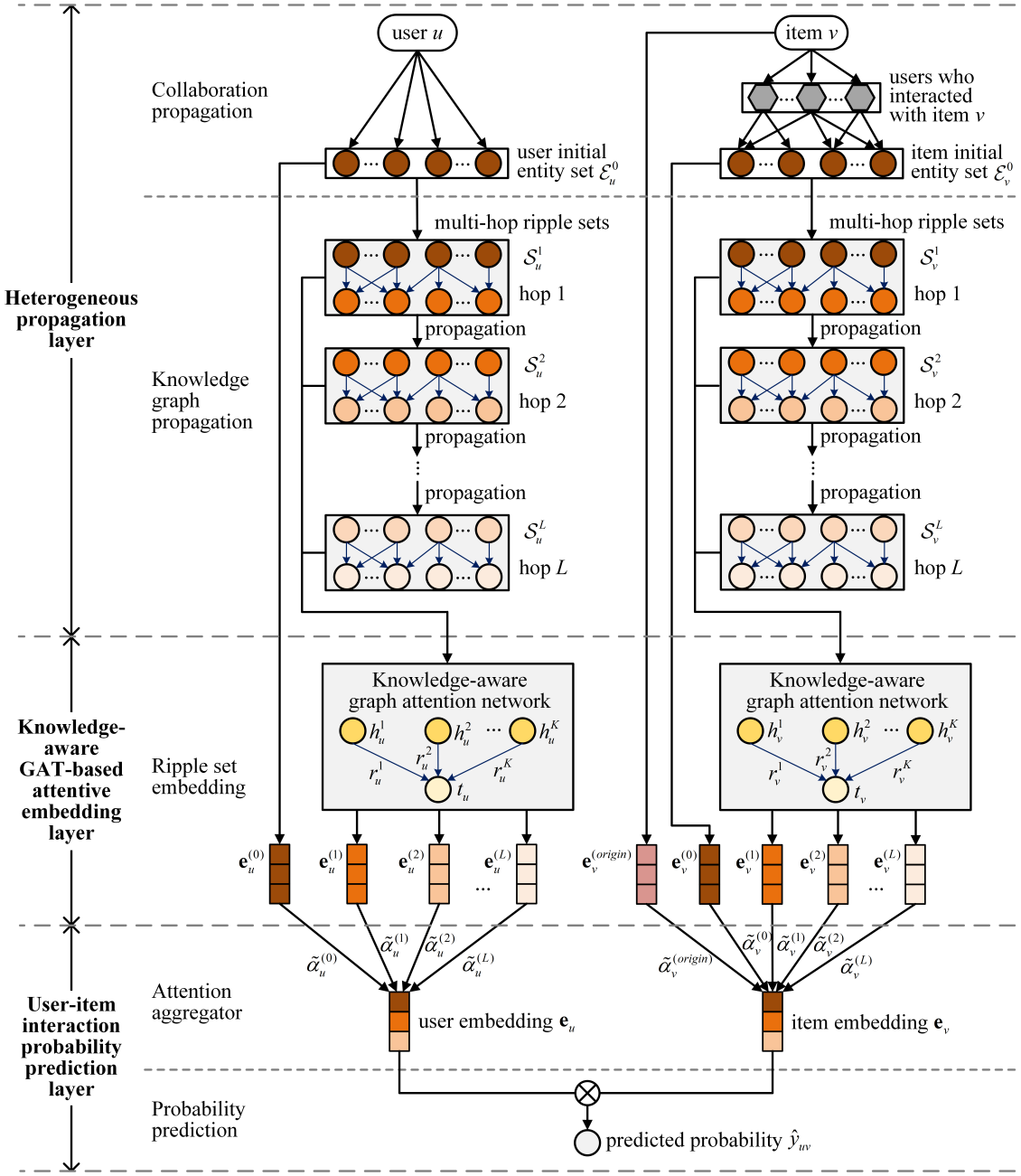

Figure 1. The overall framework of CKGAT.

- **Heterogeneous propagation layer.** This layer is composed of the collaboration propagation module and the knowledge graph propagation module. The first module propagates collaborative signals through the user-item interactions to obtain the user initial entity set and the item initial entity set. On the basis of the two initial entity sets, the second module propagates knowledge associations along the links in the knowledge graph to obtain the user's multi-hop ripple sets and the item's multi-hop ripple sets. Then, this layer outputs these sets to the next layer;
- **Knowledge-aware GAT-based attentive embedding layer.** For each input ripple set of the user/item, this layer uses a knowledge-aware graph attention network to capture the topological proximity structures of the entities (i.e., items) in the ripple set to learn the high-order entity representations, thereby generating the ripple set embedding (i.e., vector). This layer also generates the user initial entity set embedding, the item initial entity set embedding, and the original representation (i.e., vector) of the item. Finally, all these embeddings (vectors) are output to the next layer;

- **User-item interaction probability prediction layer.** For the input embeddings, this layer uses the attention aggregator to learn the weight of each embedding, and performs weighted aggregation on these embeddings to generate the user embedding and the item embedding. The two embeddings are then used to calculate the predicted probability that the user will interact with the item.

4.2. Heterogeneous Propagation Layer

As shown in Figure 1, the heterogeneous propagation layer of CKGAT uses the same heterogeneous propagation strategy as in CKAN [4] to obtain the user's multi-hop ripple sets and the item's multi-hop ripple sets. The knowledge-based high-order interaction information of the user and the item is encoded in these multi-hop ripple sets. This layer is composed of the collaboration propagation module and the knowledge graph propagation module. The former explicitly encodes the collaborative signals in the user-item interactions into the user representation and the item representation to form the first-order interaction information, whereas the latter uses the knowledge associations in the knowledge graph to expand the user representation and the item representation based on the first-order interaction information, so as to form the high-order interaction information.

4.2.1. Collaboration Propagation Module

The goal of the collaboration propagation module in CKGAT is to obtain the user initial entity set and the item initial entity set as initial seeds in the process of knowledge propagation. An item that a user has interacted with can represent the user preference to a certain extent. Therefore, for a user $u \in \mathcal{U}$, the items which the user has interacted with and are in the alignment set $\mathcal{A}$ can be used as the initial entities of the user. The user initial entity set $\mathcal{E}_u^0$ is composed of these initial entities, which is defined in Equation (1) [4].

$$\mathcal{E}_u^0 = \{e \mid (v, e) \in \mathcal{A} \text{ and } v \in \{v \mid y_{uv} = 1\}\} \tag{1}$$

Similarly, the users who interacted with the same item (essentially, all the items that these users interacted with) can also contribute to the item's feature representation as these users have similar preferences. Therefore, for an item $v \in \mathcal{V}$, the collaboration propagation module first obtains all the users who interacted with item v. Then, all the items that these users interacted with form the collaborative item set $\mathcal{V}_v$ of v, which is defined in Equation (2) [4].

$$\mathcal{V}_v = \{v_u \mid u \in \{u \mid y_{uv} = 1\} \text{ and } y_{uv_u} = 1\} \tag{2}$$

Furthermore, this module filters out the items in $\mathcal{V}_v$ that cannot be aligned with the entities in the knowledge graph through the alignment set $\mathcal{A}$ to obtain the item initial entity set $\mathcal{E}_v^0$, which is defined in Equation (3) [4].

$$\mathcal{E}_v^0 = \{e \mid (v_u, e) \in \mathcal{A} \text{ and } v_u \in \mathcal{V}_v\} \tag{3}$$

The collaboration propagation module explicitly encodes the first-order interaction information into the user/item initial entity set, thereby enhancing the user representations and item representations.

4.2.2. Knowledge Graph Propagation Module

Similar to the method in [4,23], the knowledge graph propagation module in CKGAT uses the obtained initial seeds to propagate knowledge associations along the links in the knowledge graph from near to distant, thereby obtaining the user's multi-hop ripple sets and the item's multi-hop ripple sets. These multi-hop ripple sets contain the user's extended entity sets and the item's extended entity sets, respectively. Formally, we use a uniform placeholder o to represent the symbol of user $u \in \mathcal{U}$ or item $v \in \mathcal{V}$, and use $\mathcal{E}_o^l$ to represent the extended entity set of u or v. $\mathcal{E}_o^l$ is composed of the tail entities in the triples

with $h \in \mathcal{E}_o^{l-1}$ as the head entities in the knowledge graph $\mathcal{G}$, which is defined recursively in Equation (4).

$$\mathcal{E}_o^l = \left\{ t \mid (h,r,t) \in \mathcal{G} \text{ and } h \in \mathcal{E}_o^{l-1} \right\}, l = 1,2,\ldots,L \tag{4}$$

where l represents the hop number away from the initial seeds (entities), and the hyperparameter L is the maximum hop number of the multi-hop ripple sets.

On the basis of the two extended entity sets, the knowledge graph propagation module can form the user's multi-hop ripple sets and the item's multi-hop ripple sets, in which the knowledge-based high-order interaction information of the user and the item is encoded. Formally, the l-th hop ripple set $\mathcal{S}_o^l$ is constructed by all the triples whose head entities are in the extended entity set $\mathcal{E}_o^{l-1}$, as defined in Equation (5).

$$\mathcal{S}_o^l = \left\{ (h,r,t) \mid (h,r,t) \in \mathcal{G} \text{ and } h \in \mathcal{E}_o^{l-1} \right\}, l = 1,2,\ldots,L \tag{5}$$

4.3. Knowledge-Aware GAT-Based Attentive Embedding Layer

As aforementioned, to generate the ripple set embeddings, the RippleNet framework [23] adopts the user preference propagation module, whereas the CKAN method [4] adopts the attentive network. However, these methods ignore the complex relations between entities in the multi-hop ripple sets, leading to inaccurate ripple set embeddings. This will affect the accuracy of the user/item embedding, and ultimately will affect the accuracy of the captured user's potential interest. In order to more accurately capture the user's potential interest, CKGAT adopts a knowledge-aware graph attention network to generate the ripple set embeddings. More specifically, on the basis of the graph attention network [35], we construct the knowledge-aware graph attention network by additionally exploiting the embeddings of the relations between entities.

As shown in Figure 1, the knowledge-aware GAT-based attentive embedding layer uses the knowledge-aware graph attention network to capture the topological proximity structures of the entities in the multi-hop ripple sets and learn the high-order entity representations, thereby generating the ripple set embeddings. More specifically, this layer consists of the following four steps.

The first step is to obtain multiple neighboring head entities of the tail entity in each triple from the user's or the item's multi-hop ripple sets, thereby forming the neighboring entity sets. Assuming that the input of the knowledge-aware graph attention network is the ripple set $\mathcal{S}_o^l, l = 1,2,\ldots,L$, where the unified placeholder o represents the user u or the item v, and the corresponding extended entity set of $\mathcal{S}_o^l$ is $\mathcal{E}_o^l$. The knowledge-aware graph attention network acquires the K neighboring head entities of each tail entity $t_o \in \mathcal{E}_o^l$ of each triple in $\mathcal{S}_o^l$ to form the neighboring entity set $\mathcal{N}(t_o)$, with the hyperparameter $K = |\mathcal{N}(t_o)|$. The neighboring entity set is defined in Equation (6).

$$\mathcal{N}(t_o) = \left\{ h_o^k \mid h_o^k \in \mathcal{E}_o^{l-1} \text{ and } \left(h_o^k, r_o^k, t_o\right) \in \mathcal{S}_o^l \right\}, k = 1,2,\ldots,K \tag{6}$$

The second step is to learn the neighborhood representation of the tail entity $t_o \in \mathcal{E}_o^l$ based on the obtained neighboring entity set. As shown in Figure 2, let the embedding of each neighboring head entity $h_o^k \in \mathcal{N}(t_o)$ be $\mathbf{e}_{h_o^k}, k = 1,2,\ldots,K$. The knowledge-aware graph attention network first uses the triple $\left(h_o^k, r_o^k, t_o\right) \in \mathcal{S}_o^l$ to learn a score π^k, and normalize the score to obtain the weight $\tilde{\pi}^k$. Then, the network performs weighted aggre-

gation on the embedding $\mathbf{e}_{h_o^k}$ of each neighboring head entity to obtain the neighborhood representation $\mathbf{e}_{\mathcal{N}(t_o)}$ of the entity t_o. The above process is defined in Equation (7).

$$\begin{aligned} \pi^k &= \text{LeakyReLU}\left(\mathbf{w}_1^\top \left[\mathbf{W}_1\mathbf{e}_{h_o^k} || \mathbf{W}_1\mathbf{r}_o^k || \mathbf{W}_1\mathbf{e}_{t_o}\right]\right) \\ \tilde{\pi}^k &= \frac{\exp\left(\pi^k\right)}{\sum_{h_o^{k'} \in \mathcal{N}(t_o)} \exp\left(\pi^{k'}\right)} \\ \mathbf{e}_{\mathcal{N}(t_o)} &= \sum_{k=1}^{K} \tilde{\pi}^k \mathbf{e}_{h_o^k} \end{aligned} \tag{7}$$

where LeakyReLU is a type of activation function based on a ReLU; this activation function can assign a non-zero slope to all negative values. The parameters $\mathbf{w}_1 \in \mathbb{R}^{3d}$ and $\mathbf{W}_1 \in \mathbb{R}^{d\times d}$ are a weight vector and a weight matrix, respectively, that are determined through parameter learning. $\mathbf{e}_{h_o^k}, \mathbf{r}_o^k, \mathbf{e}_{t_o} \in \mathbb{R}^d$ represent the d-dimensional embedding (vector) of entity h_o^k, relation r_o^k, and entity t_o, respectively. $\|$ denotes a concatenation operation.

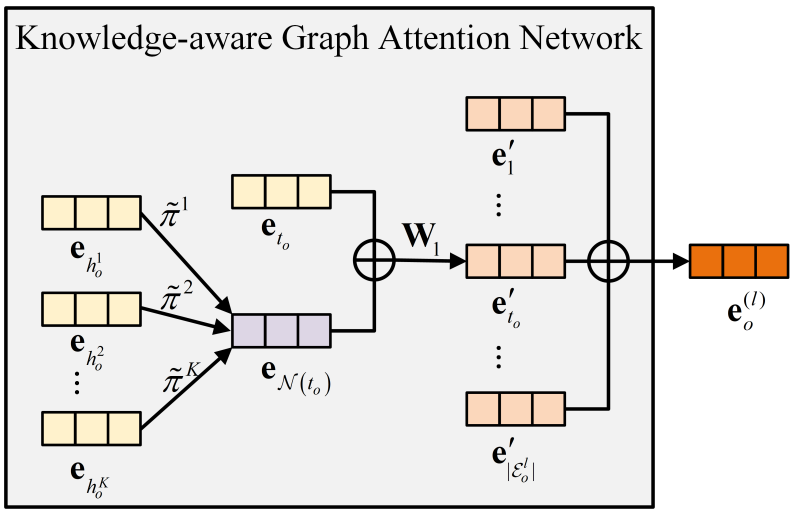

Figure 2. Knowledge-aware graph attention network.

In the third step, the graph attention network combines the obtained neighborhood representation $\mathbf{e}_{\mathcal{N}(t_o)}$ with the embedding $\mathbf{e}_{t_o}$ of entity t_o to generate a high-order representation $\mathbf{e}'_{t_o}$ of t_o, which is defined in Equation (8).

$$\mathbf{e}'_{t_o} = \sigma\left(\mathbf{W}_1\left(\mathbf{e}_{t_o} + \mathbf{e}_{\mathcal{N}(t_o)}\right)\right) \tag{8}$$

where $\sigma(\cdot)$ is the sigmoid activation function.

In the fourth step, the graph attention network obtains the embedding $\mathbf{e}_o^{(l)}$ of the ripple set $\mathcal{S}_o^l$ by aggregating the high-order representation of each entity $t_o \in \mathcal{E}_o^l$, which is defined in Equation (9).

$$\mathbf{e}_o^{(l)} = \sum_{t_o \in \mathcal{E}_o^l} \mathbf{e}'_{t_o}, l = 1, 2, \ldots, L \tag{9}$$

In this way, L ripple set embeddings of the user and L ripple set embeddings of the item are obtained, respectively.

Finally, since the entities in the user initial entity set have a strong connection with the user, and the entities in the item initial entity set have a strong connection with the item, the knowledge-aware GAT-based attentive embedding layer also needs to use the user initial entity set embedding $\mathbf{e}_u^{(0)}$ to enrich the user embedding, and use the item initial

entity set embedding $\mathbf{e}_v^{(0)}$ and the original item representation (vector) $\mathbf{e}_v^{(origin)}$ to enrich the item embedding. This way, the user representation sets $\mathcal{T}_u$ and the item representation sets $\mathcal{T}_v$ are formed, which are defined in Equation (10).

$$\begin{aligned} \mathcal{T}_u &= \left\{ \mathbf{e}_u^{(0)}, \mathbf{e}_u^{(1)}, \ldots, \mathbf{e}_u^{(L)} \right\} \\ \mathcal{T}_v &= \left\{ \mathbf{e}_v^{(origin)}, \mathbf{e}_v^{(0)}, \mathbf{e}_v^{(1)}, \ldots, \mathbf{e}_v^{(L)} \right\} \end{aligned} \tag{10}$$

Using the same calculation method in [4], the user/item initial entity set embedding $\mathbf{e}_o^{(0)}$ is defined in Equation (11).

$$\mathbf{e}_o^{(0)} = \frac{\sum_{e \in \varepsilon_o^0} \mathbf{e}}{\left| \mathcal{E}_o^0 \right|} \tag{11}$$

As in [4], the embedding of the entity aligned with the item $v \in \mathcal{V}$ in the knowledge graph $\mathcal{G}$ can be used as the original representation $\mathbf{e}_v^{(origin)}$ of v, as defined in Equation (12).

$$\mathbf{e}_v^{(origin)} = \frac{\sum_{e \in \{e \mid (e,v) \in \mathcal{A}\}} \mathbf{e}}{|\{e \mid (e,v) \in \mathcal{A}\}|} \tag{12}$$

4.4. User-Item Interaction Probability Prediction Layer

This layer in CKGAT uses the user representation set $\mathcal{T}_u$ and the item representation set $\mathcal{T}_v$, which are uniformly denoted as $\mathcal{T}_o$, to respectively generate the user embedding and the item embedding, thereby calculating the predicted probability that the user will interact with the item. Inspired by the attention mechanism [36–38], for the input user/item representation set $\mathcal{T}_o$, the prediction layer uses the attention aggregator to generate the user/item embedding $\mathbf{e}_o$. Specifically, the attention aggregator first learns an attention score $\alpha_o^{(i)}$ for each embedding $\mathbf{e}_o^{(i)} \in \mathcal{T}_o$. Then, it normalizes the score to obtain the weight coefficient $\tilde{\alpha}_o^{(i)}$ of the embedding $\mathbf{e}_o^{(i)}$. Finally, the attention aggregator performs weighted aggregation on all $\mathbf{e}_o^{(i)}$ to obtain the user/item embedding $\mathbf{e}_o$. The above process is defined in Equation (13).

$$\begin{aligned} \alpha_o^{(i)} &= \mathbf{w}_2^\top \tanh\left(\mathbf{W}_2 \mathbf{e}_o^{(i)} \right) \\ \tilde{\alpha}_o^{(i)} &= \frac{\exp\left(\alpha_o^{(i)} \right)}{\sum_{i'=1}^{|T_o|} \exp\left(\alpha_o^{(i')} \right)} \\ \mathbf{e}_o &= \sigma\left(\mathbf{W}_3 \sum_{e_o^{(i)} \in \mathcal{T}_o} \tilde{\alpha}_o^{(i)} \mathbf{e}_o^{(i)} + \mathbf{b} \right) \end{aligned} \tag{13}$$

where tanh is an activation function which endows the prediction model with nonlinearity. Parameters $\mathbf{w}_2 \in \mathbb{R}^d$ and $\mathbf{W}_2, \mathbf{W}_3 \in \mathbb{R}^{d \times d}$ are the weight vector and the weight matrices, respectively. $\mathbf{b} \in \mathbb{R}^d$ is the bias. These parameters are determined through parameter learning.

After obtaining the user embedding $\mathbf{e}_u$ and the item embedding $\mathbf{e}_v$, CKGAT calculates the predicted probability $\hat{y}_{uv}$ that the user u will interact with the item v by performing the inner product operation on the two embeddings, as defined in Equation (14).

$$\hat{y}_{uv} = \mathbf{e}_u^\top \mathbf{e}_v \tag{14}$$

4.5. Model Learning and Recommendation Generation

CKGAT adopts the negative sampling strategy to improve the efficiency of model learning, and the number of negative samples is the same as that of positive ones for each user. As in CKAN [4], the loss function of CKGAT is defined in Equation (15).

$$\mathcal{L} = \sum_{u \in \mathcal{U}} \left(\sum_{v \in \{v|(u,v) \in \mathcal{P}^+\}} \mathcal{J}(y_{uv}, \hat{y}_{uv}) - \sum_{v \in \{v|(u,v) \in \mathcal{P}^-\}} \mathcal{J}(y_{uv}, \hat{y}_{uv}) \right) + \lambda \|\Theta\|_2^2 \quad (15)$$

where $\mathcal{J}(y_{uv}, \hat{y}_{uv})$ is the cross-entropy loss. $\mathcal{P}^+$ represents the positive user-item pair set, $\mathcal{P}^-$ is the opposite. $\Theta = \{\mathbf{E}, \mathbf{R}, \mathbf{w}_1, \mathbf{w}_2, \mathbf{b}, \mathbf{W}_i, \forall i \in \{0, 1, 2\}\}$ represents the set of model parameters, where $\mathbf{E}$ and $\mathbf{R}$ are the embedding table of all entities and the embedding table of all relations, respectively. $\|\Theta\|_2^2$ is the L2-regularizer. Hyper-parameter λ is used to balance the regularizer.

CKGAT uses the stochastic gradient descent (SGD) algorithm to minimize the above loss function to learn all the parameters of CKGAT. Once the learning process is completed, CKGAT can obtain the trained user embedding and item embedding. Given a target user, CKGAT first uses the user embedding and the embeddings of all candidate items that the user has not interacted with to calculate the predicted probabilities of the items. Then the predicted probabilities are sorted in descending order, thereby generating a top-N recommendation list for the user.

5. Experiments

In this section, we describe our experiments, purposes of which are answering the following research questions (RQs).

(1) RQ1: Does our CKGAT method outperform the state-of-the-art methods in the field of knowledge graph-based recommendation in terms of recommendation accuracy?

(2) RQ2: Is CKGAT superior to the current state-of-the-art propagation-based recommendation methods in terms of recommendation diversity?

(3) RQ3: How do the different components of CKGAT (i.e., the knowledge-aware GAT-based attentive embedding layer and the attention aggregator) influence the recommendation accuracy of the method?

(4) RQ4: How do different hyperparameter settings affect the CKGAT method?

5.1. Experimental Datasets

Our experiments used the following four real-world datasets, Last.FM, Book-Crossing, MovieLens 20M, and Dianping-Food, as follows.

- **Last.FM** [39]**.** This dataset contains social networking, tagging, and music artist listening information from a set of 2000 users from the Last.fm online music system;
- **Book-Crossing** [40]**.** This dataset collects explicit ratings (ranging from 0 to 10) from different readers about various books in the book-crossing community;
- **MovieLens 20M** [41]**.** This dataset is a widely used benchmark dataset in movie recommendation, which contains approximately 20 million explicit user ratings for movies (ranging from one to five) on the MovieLens website;
- **Dianping-Food** [42]**.** This dataset is provided by Dianping.com, which contains 10 million interaction data (including clicks and purchases, etc.) between approximately 2 million users and 1000 restaurants.

Instead of the above four original datasets, our experiments directly used the preprocessed Last.FM, Book-Crossing, and MovieLens 20M datasets, and their corresponding knowledge graphs released on GitHub [43] by Wang Z. et al. [4]. We also used the preprocessed Dianping-Food dataset and its corresponding knowledge graph released on GitHub [44] by Wang H. et al. [7]. Table 1 shows the statistics of the four experimental datasets and their corresponding knowledge graphs. As in [4,7], each dataset was randomly divided into a training set, a validation set, and a test set in a ratio of 6:2:2. Furthermore, we

followed the methods in [4,7] to randomly select 100 users from the test set and generate a top-N recommendation list for each user.

Table 1. Statistics of the experimental datasets and their corresponding knowledge graphs.

Datasets		Last.FM	Book-Crossing	MovieLens 20M	Dianping-Food
user-item interactions	#users	1872	17,860	138,159	2,298,698
	#items	3846	14,967	16,954	1362
	#interactions	42,346	139,746	13,501,622	23,416,418
knowledge graph	#entities	9366	77,903	102,569	28,115
	#relations	60	25	32	7
	#triples	15,518	151,500	499,474	160,519

5.2. Comparison Methods

In order to evaluate the effectiveness of our CKGAT method, we chose experimental comparison methods according to the following three rules.

- In accordance with the practice of choosing experimental comparison methods in the existing research work [2–8,15,21,23,24,28–30,33,34] in the field, we chose one classical collaborative filtering method, one typical embedding-based recommendation method, and six representative propagation-based recommendation methods (as the mainstream methods of knowledge graph-based recommendation);
- Unlike the above-mentioned research works, which did not choose a connection-based recommendation method as an experimental comparison method, we chose KPRN, a typical connection-based recommendation method, as the comparison method;
- We excluded recommendation methods whose codes were not available at the time of writing this paper, such as the methods in [24,28–30,33,34].

As a consequence, CKGAT was compared with nine representative methods which are divided into four categories: collaborative filtering method (BPRMF), embedding-based recommendation method (CKE), connection-based recommendation method (KPRN), and propagation-based recommendation methods which are further divided into the user representation-refinement approaches (RippleNet and CKAN), the item representation-refinement approaches (KGCN and KGNN-LS), and both user and item representation-refinements approaches (KGAT and KGIN). Below are brief descriptions of these comparison methods.

- **BPRMF** [45]**.** This method is a Bayesian personalized ranking (BPR) optimized matrix factorization (MF) model achieved by applying LearnBPR to MF.
- **CKE** [2]**.** This method is a typical knowledge graph embedding-based recommendation method that combines the structural knowledge, textual knowledge and visual knowledge of items to learn item representations.
- **KPRN** [3]**.** This method is a typical connection-based recommendation method that generates path representations by combining the semantics of entities and relations and distinguishes the importance of different paths, thereby capturing user preferences.
- **RippleNet** [23]**.** This method is a classical propagation-based recommendation method that enhances user representations by propagating users' potential preferences in the knowledge graph.
- **CKAN** [4]**.** This method belongs to the propagation-based recommendation methods that uses a heterogeneous propagation strategy and an attention network to learn ripple set embeddings, thereby generating user embeddings and item embeddings.
- **KGCN** [5]**.** This method belongs to the propagation-based recommendation methods, which applies the graph convolutional network to the knowledge graph to aggregate neighborhood information to refine item representations.

- **KGNN-LS** [7]**.** This method belongs to the propagation-based recommendation methods that adds a label-smoothness mechanism to the KGCN framework to propagate user-interaction labels, so as to provide effective recommendations.
- **KGAT** [8]**.** This method belongs to the propagation-based recommendation methods that applies the graph attention network to the collaborative knowledge graph to learn user representations and item representations.
- **KGIN** [6]**.** This method is currently the state-of-the-art propagation-based recommendation method. It uses auxiliary item knowledge to explore the users' intention behind the user-item interactions, thus refining the representations of users and items.

Our experiments directly used the Python codes released on GitHub by the original publications of the methods (or a third party) to implement these comparison methods. More specifically, the codes of BPRMF, CKE and KGAT were from the GitHub repository knowledge_graph_attention_network [46]. The code of KPRN was from the GitHub repository KPRN [47]. The code of RippleNet was from the GitHub repository RippleNet [48]. The code of CKAN was from the GitHub repository CKAN [49]. The code of KGCN was from the GitHub repository KGCN [50]. The code of KGNN-LS was from the GitHub repository KGNN-LS [51]. The code of KGIN was from the GitHub repository Knowledge_Graph_based_intent_Network [52].

In essence, our CKGAT method makes two important improvements to CKAN [4]: the generation of ripple-set embeddings, and the generation of user/item embeddings. When implementing our CKGAT method, we used the PyTorch framework to modify the CKAN code in two major aspects: (1) Modify the knowledge-aware attentive embedding layer in CKAN to the knowledge-aware GAT-based attentive embedding layer in CKGAT; (2) Modify the aggregator in CKAN to the attention aggregator in CKGAT. Therefore, the CKAN method can be regarded as a basic method of our CKGAT method. The Python code of our CKGAT method is released on the GitHub repository CKGAT at https://github.com/hu-dske/CKGAT (accessed on 29 December 2021).

5.3. Hyperparameter Settings

During the experiments, the hyperparameters for the nine comparison methods were set in the same way as in the original publications of the comparison methods.

The hyperparameters for our CKGAT method were optimized via grid search on the four experimental datasets. More specifically, the hyperparameter ranges for grid search were as follows: the embedding dimension d in {8, 16, 32, 64, 128, 256}, the learning rate δ of the stochastic gradient descent algorithm in $\{10^{-3}, 5 \times 10^{-3}, 10^{-2}, 5 \times 10^{-2}, 10^{-1}\}$, the $L2$ regularization coefficient λ in $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$, the maximum hop number L of ripple sets in {1, 2, 3, 4}, the number K of neighboring entities in {2, 3, 4, 5}. Finally, in the following experiments, d was set to 64, δ was set to 5×10^{-3}, and λ was set to 10^{-5} for the four datasets. The hyperparameters L and K were obtained through the hyperparameter sensitivity experiment as described in Section 5.4.4. As a result, in the following experiments we set $L = 3$ and $K = 3$ for the Last.FM dataset, $L = 2$ and $K = 4$ for the Book-Crossing dataset, as well as $L = 1$ and $K = 5$ for the MovieLens 20M, and Dianping-Food datasets.

5.4. Experimental Results

5.4.1. Recommendation Accuracy (RQ1)

Like the related work [4,5,7,8] in the recommendation field, our experiments used four popular accuracy metrics [53], Precision@N, Recall@N, F1-measure@N, and NDCG@N, with N = {5, 10, 20, 50, 100}, to evaluate the accuracy of the recommendation methods. Figures 3–6 show the top-N recommendation accuracy comparisons between our CKGAT method and the nine comparison methods on the four experimental datasets. We have the following observations from the results:

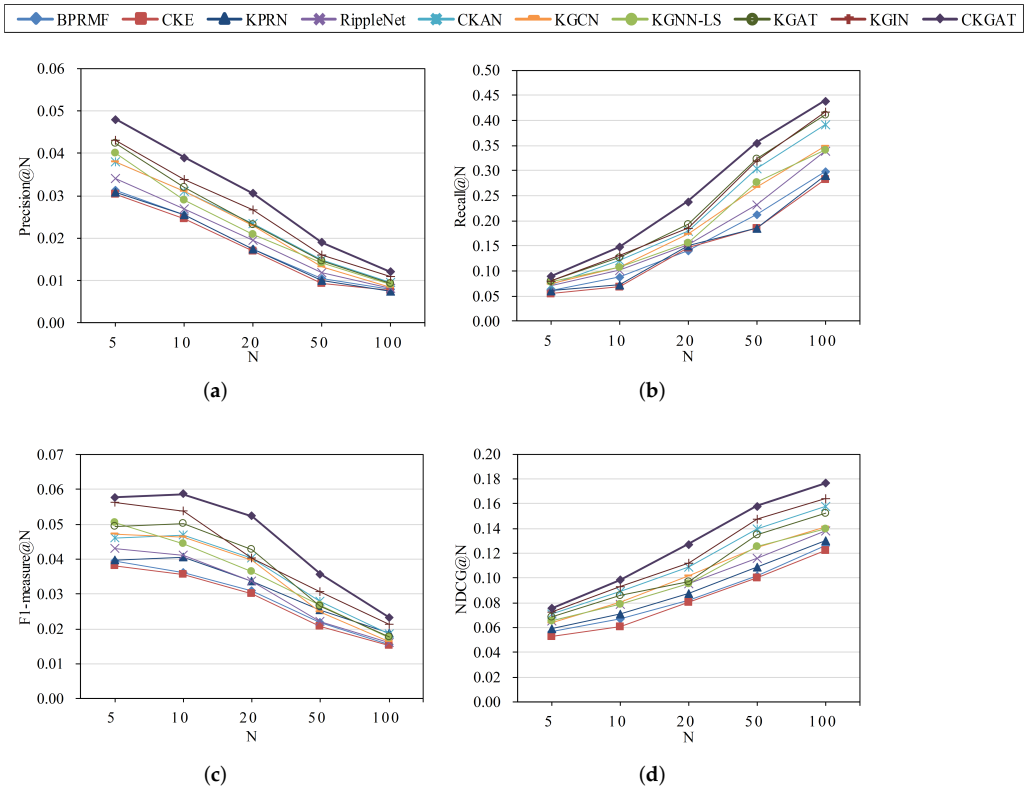

Figure 3. Top-N recommendation performance comparison on the Last.FM dataset. (**a**) Precesion@N; (**b**) Recall@N; (**c**) F1-measure@N; (**d**) NDCG@N.

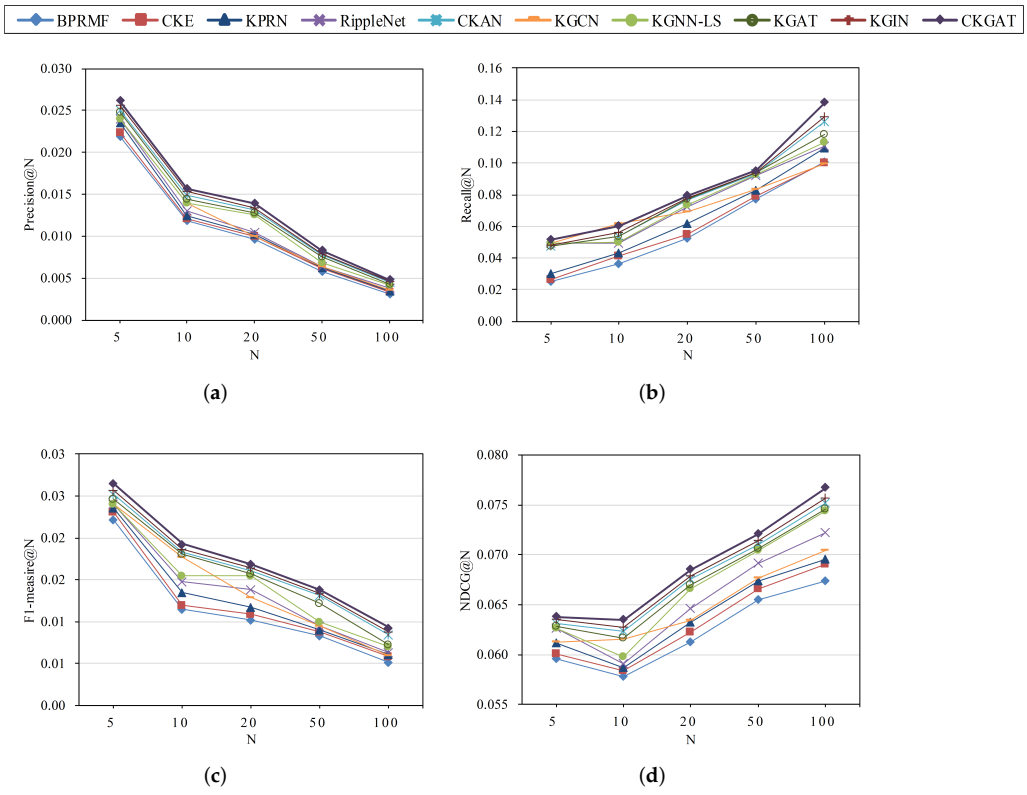

Figure 4. Top-N recommendation performance comparison on the Book-Crossing dataset. (**a**) Precesion@N; (**b**) Recall@N; (**c**) F1-measure@N; (**d**) NDCG@N.

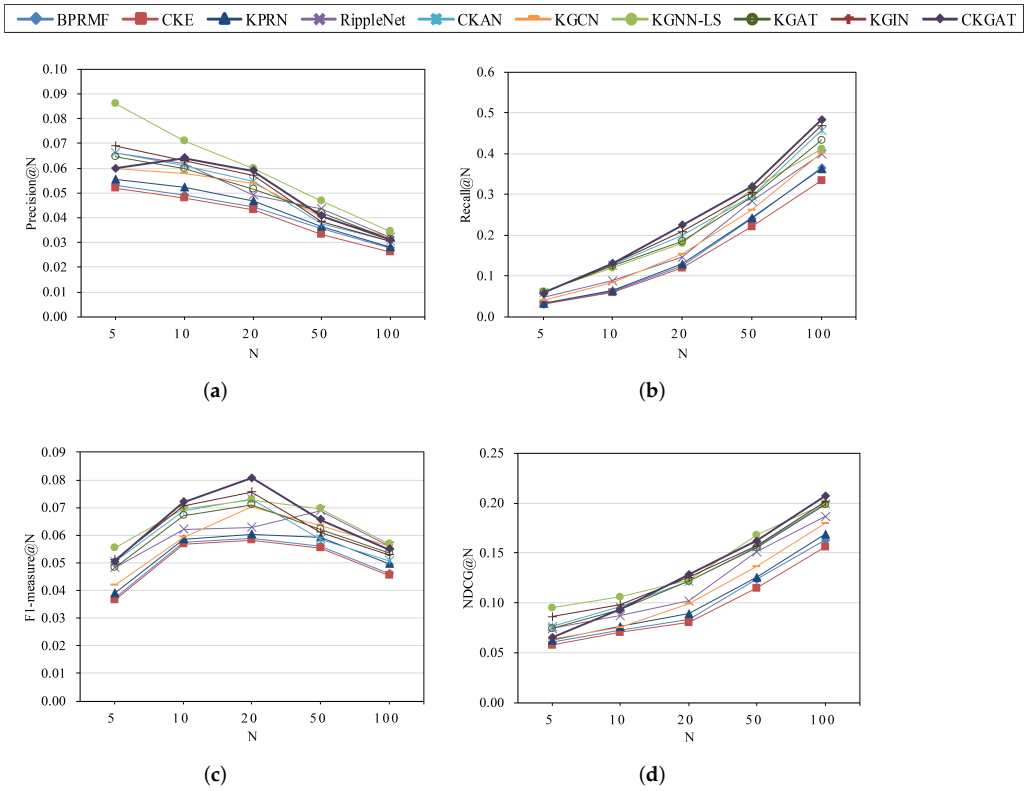

Figure 5. Top-N recommendation performance comparison on the MovieLens 20M dataset. (**a**) Precision@N; (**b**) Recall@N; (**c**) F1-measure@N; (**d**) NDCG@N.

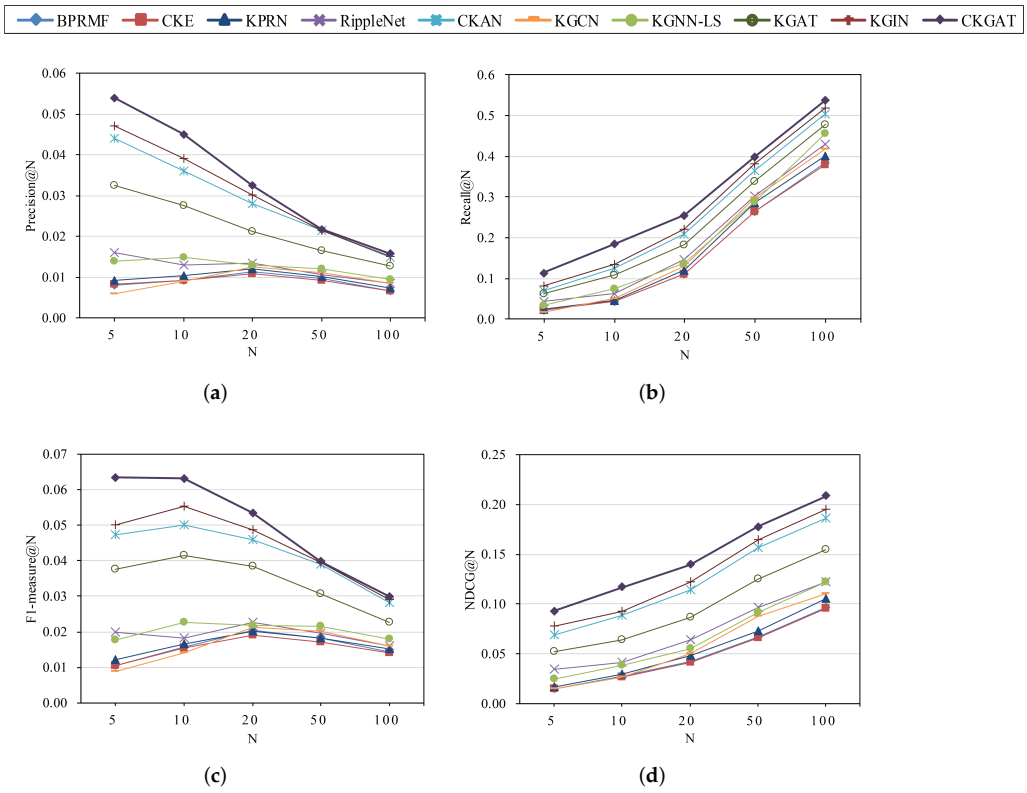

Figure 6. Top-N recommendation performance comparison on the Dianping-Food dataset. (**a**) Precision@N; (**b**) Recall@N; (**c**) F1-measure@N; (**d**) NDCG@N.

1. CKGAT achieves the best recommendation accuracy across all evaluation metrics on the four datasets, with the exception of the precision metric on the MovieLens 20M dataset. This result shows that on the basis of the heterogeneous propagation strategy, the user embeddings and the item embeddings generated by the knowledge-aware GAT-based attentive embedding layer and the attention aggregator enable CKGAT to accurately capture the users' potential interests and improve the accuracy of personalized recommendation.
2. In terms of the precision metric on the MovieLens 20M dataset, CKGAT is slightly lower than KGNN-LS, but is comparable with KGNN-LS in terms of Precision@20. Since N is usually set to 20–50 [54] in most practical application scenarios, it can be considered that CKGAT still has good recommendation accuracy on the MovieLens 20M dataset. The reason why the above exception exists may be that the average number of user-item interactions per user is so large that the propagation of knowledge in the knowledge graph is almost ineffective.
3. The recommendation accuracy of CKGAT on all the datasets is significantly better than CKAN. This result indicates that the topological proximity structures of entities in multi-hop ripple sets can effectively enrich the ripple set embeddings, thereby generating the refined user embeddings and item embeddings.
4. We have observed that the recommendation accuracy of CKGAT on all the datasets is overall better than all the comparison methods, and the recommendation accuracy of CKAN on all the datasets is overall better than other comparison methods except KGIN. These results show that both CKGAT and CKAN, which adopt the heterogeneous propagation strategy, can enhance the user embeddings and the item embeddings by effectively combining the collaborative signals in the user-item interactions and the knowledge associations in the knowledge graph, thereby improving the recommendation performance.
5. CKGAT, KGIN, CKAN, and KGAT outperform KGNN-LS, KGCN, and RippleNet in terms of most evaluation metrics on all the datasets. This result shows that CKGAT, KGIN, CKAN, and KGAT can use both the first-order user-item interaction information in the user-item interaction matrix and the knowledge associations in the knowledge graph to mine accurate user preferences.
6. The seven propagation-based recommendation methods, including CKGAT, are significantly better than the baseline methods BPRMF, CKE and KPRN across all evaluation metrics on all the datasets. This result shows that the propagation-based methods can effectively exploit the high-order relations in the knowledge graph to more accurately capture the users' potential interests in items.

5.4.2. Recommendation Diversity (RQ2)

The notion of diversity means that the set of items within a single recommended list should be as diverse as possible [53]. Even if the accuracy of a recommendation method is high, it may not be able to generate diversified recommendations for users. The diversity of recommended results can prevent users from getting bored because of many non-diversified items being recommended. Therefore, the diversity of recommendation results is also an important evaluation aspect to measure recommendation performance. Our experiment used the DIV@N metric [55] to evaluate the diversity of recommendation results. The total number of all possible recommendation pairs $(\mathcal{R}_i, \mathcal{R}_j)$, $i \neq j$ in M top-N recommendations is $M(M-1)/2$, the overlap rate of a recommendation pair is $\left|R_i \cap R_j\right| / \left|R_i \cup R_j\right|$. The DIV@N metric measures the mean non-overlap ratio of all recommendation pairs, which is defined in Equation (16) [55].

$$\mathrm{DIV@N} = \frac{2}{M(M-1)} \sum_{i,j \in \{1,2,\ldots,M\}, i \neq j} \left(1 - \frac{\left|R_i \cap R_j\right|}{\left|R_i \cup R_j\right|}\right) \tag{16}$$

where $M = 100$ because we randomly generate top-N recommendations for 100 users during the test.

Figure 7 shows the DIV@10 and DIV@20 results of CKGAT and the four propagation-based methods (KGNN-LS, KGAT, CKAN and KGIN) on the Movielens 20M dataset. We have the following observations from the results:

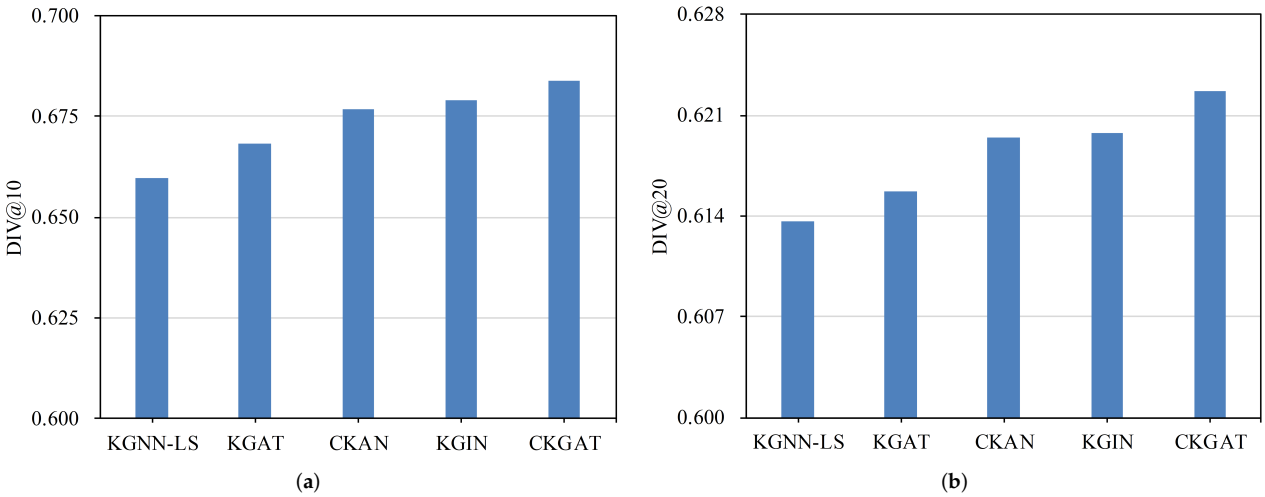

Figure 7. Recommendation diversity performance comparison on the MovieLens 20M dataset. (**a**) DIV@10; (**b**) DIV@20.

1. The diversity of CKGAT's recommendation results is significantly better than that of KGNN-LS, KGAT, CKAN, and KGIN, although the recommendation accuracy of CKGAT on the MovieLens 20M dataset is slightly better than these four methods.
2. The diversity of KGIN's recommendation results is slightly better than that of CKAN, KGAT and KGNN-LS. This may be due to the fact that KGIN handles user intent at a more fine-grained level, and considers multiple aspects of user intent.
3. The diversity of CKAN's recommendation results is slightly better than that of KGNN-LS and KGAT. This may be due to the fact that CKAN can extract a variety of user preferences and various item features in the process of heterogeneous propagation.
4. The diversity of KGAT's recommendation results is better than that of KGNN-LS. This may be because KGAT uses a collaborative knowledge graph containing user-item interaction information, which contains a wealth of user preference information.

5.4.3. Different Components' Influences (RQ3)

To evaluate the influences of the two components of CKGAT (i.e., the knowledge-aware GAT-based attentive embedding layer and the attention aggregator) on recommendation accuracy, we have designed the following two variant methods of CKGAT:

- **CKGAT$_{\text{w/o Att}}$.** This variant method is achieved by removing the attention aggregator from CKGAT. The purpose is to verify the influence of the knowledge-aware GAT-based attentive embedding layer on recommendation accuracy.
- **CKGAT$_{\text{w/o GAT}}$.** This variant method is achieved by removing the knowledge-aware GAT-based attentive embedding layer from CKGAT. The purpose is to verify the influence of the attention aggregator on recommendation accuracy.

Table 2 shows the top-N recommendation results of CKAN, CKGAT, and the two variant methods on the four datasets, with the best results among all the methods highlighted in bold.

Table 2. Recommendation accuracy of CKAN, CKGAT, and the two variants on the four datasets.

Datasets	Metrics	CKAN	$\text{CKGAT}_{\text{w/o Att}}$	$\text{CKGAT}_{\text{w/o GAT}}$	CKGAT
Last.FM	precision@10	0.0310	0.0353	0.0338	**0.0390**
	recall@10	0.1215	0.1383	0.1345	**0.1481**
	F1-measure@10	0.0469	0.0562	0.0530	**0.0587**
	NDCG@10	0.0893	0.0935	0.0928	**0.0984**
Book-Crossing	precision@10	0.0149	0.0152	0.0154	**0.0157**
	recall@10	0.0537	0.0562	0.0553	**0.0603**
	F1-measure@10	0.0183	0.0185	0.0186	**0.0193**
	NDCG@10	0.0623	0.0628	**0.0640**	0.0635
MovieLens 20M	precision@10	0.0610	0.0622	0.0628	**0.0640**
	recall@10	0.1273	0.1282	0.1280	**0.1319**
	F1-measure@10	0.0687	0.0692	0.0694	**0.0719**
	NDCG@10	**0.0962**	0.0923	0.0935	0.0935
Dianping-Food	precision@10	0.0360	0.0430	0.0406	**0.0450**
	recall@10	0.1254	0.1545	0.1432	**0.1855**
	F1-measure@10	0.0500	0.0566	0.0542	**0.0632**
	NDCG@10	0.0883	0.0993	0.0965	**0.1168**

We have the following observations from the results:

1. CKGAT is significantly better than the two variants $\text{CKGAT}_{\text{w/o Att}}$ and $\text{CKGAT}_{\text{w/o GAT}}$ across all evaluation metrics on all the datasets, with the exception of the NDCG@10 metric on the Book-Crossing dataset. This shows that CKGAT can more effectively capture users' preferences and improve recommendation accuracy by integrating the knowledge-aware GAT-based attentive embedding layer and the attention aggregator to learn user embeddings and item embeddings.
2. The two variant methods are superior to CKAN (the basic method of CKGAT) across all the evaluation metrics on all the datasets, with the exception of the NDCG@10 metric on the MovieLens 20M dataset. This shows that using the knowledge-aware GAT-based attentive embedding layer to capture the topological proximity structures of entities in multi-hop ripple sets as well as using the attention aggregator to distinguish the importance of ripple set embeddings can refine the representations of users and the representations of items.
3. The recommendation accuracy of $\text{CKGAT}_{\text{w/o Att}}$ is better than $\text{CKGAT}_{\text{w/o GAT}}$ in terms of most evaluation metrics on all the datasets. This shows that the topological proximity structures of the entities in multi-hop ripple sets play a more important role in learning the user representations and item representations.

5.4.4. Hyperparameter Sensitivity (RQ4)

The CKGAT method generates user/item embeddings by aggregating the ripple set embeddings. Therefore, the maximum hop number L of the ripple sets directly affects the generation of user/item embeddings, and then affects the recommendation performance. In order to verify the influence of the hyperparameter L on the recommendation performance, we analyzed the sensitivity of CKGAT to L through experiments. In the experiments, L was changed from one to four, while other hyperparameters remained fixed. The experimental results are shown in Table 3, with the best results among different parameter values on a dataset highlighted in bold.

Table 3. The Recall@10 results in terms of the maximum hop number L.

L	1	2	3	4
Last.FM	0.1042	0.1269	**0.1481**	0.1203
Book-Crossing	0.0371	**0.0603**	0.0417	0.0201
MovieLens 20M	**0.1319**	0.1072	0.0808	0.0742
Dianping-Food	**0.1855**	0.1409	0.1125	0.1061

We have the following observations from the experimental results:

1. CKGAT achieves the best recommendation performance when $L = 3$ and $L = 2$ on the Last.FM and Book-Crossing datasets, respectively, and achieves the best recommendation performance when $L = 1$ on both the MovieLens 20M and Dianping-Food datasets.
2. When L increases to four, the recommendation performance of CKGAT on the four datasets is the worst. This may be because when the maximum hop number is large, CKGAT introduces entities with lower relevance to users/items and generates inaccurate user/item embeddings, thus limiting the recommendation performance.

In addition, CKGAT learns the high-order entity representations with the help of their multiple neighboring entities in multi-hop ripple sets. Therefore, the number K of neighboring entities directly affects the learning of the entity's high-order representation and then affects the recommendation performance. In order to verify the influence of hyperparameter K on the recommendation performance, we further analyzed the sensitivity of CKGAT to K through experiments. In the experiments, K varied from two to five while other hyperparameters remained fixed. The experimental results are shown in Table 4, with the best results among different parameter values on a dataset highlighted in bold.

Table 4. The Recall@10 results in terms of the number of neighboring entities K.

K	2	3	4	5
Last.FM	0.1173	**0.1481**	0.1071	0.0762
Book-Crossing	0.0208	0.0298	**0.0603**	0.0450
MovieLens 20M	0.0477	0.0672	0.1014	**0.1319**
Dianping-Food	0.0570	0.1257	0.1345	**0.1855**

We have the following observations from the experimental results:

1. CKGAT achieves the best recommendation performance when $K = 3$ and $K = 4$ on the Last.FM and Book-Crossing datasets, respectively, and achieves the best recommendation performance when $K = 5$ on both the MovieLens 20M and Dianping-Food datasets.
2. When K decreases to two, the recommendation performance of CKGAT on the four datasets is the worst. This may be because when the number of neighbors is small, the high-order entity representations learned by CKGAT from the topological proximity structures are insufficient to support the generation of accurate user/item embeddings, thus limiting the recommendation performance.

6. Conclusions

Knowledge graph-based recommendation methods are a hot research topic in the field of recommender systems. The propagation-based recommendation methods are mainstream knowledge graph-based recommendation methods, but they usually ignore the complex relations between entities in the multi-hop ripple sets and do not distinguish the importance of different ripple sets, resulting in inaccurate user potential interests being captured. To overcome these shortcomings, this paper proposes a top-N recommendation method named collaborative knowledge-aware graph attention network (CKGAT). This method can learn refined ripple set embeddings, thereby generating accurate user

embeddings and item embeddings, so as to accurately capture users' potential interests in items. The extensive experiments on four real-world datasets have demonstrated that the proposed CKGAT method outperforms the state-of-the-art methods for knowledge graph-based recommendation, in terms of recommendation accuracy and diversity.

Our future work will focus on investigating how to extend the method to exploit the relations between users and the user information contained in social networks, so as to provide more accurate recommendations.

Author Contributions: Conceptualization, Z.X. and J.L.; methodology, Z.X. and J.L.; software, J.L. and H.L.; validation, J.L. and H.L.; formal analysis, Z.X., J.L. and H.L.; investigation, J.L. and H.L.; resources, Z.X. and Y.T.; data curation, J.L., H.L. and Q.Z.; writing—original draft preparation, J.L. and Z.X.; writing—review and editing, Z.X., H.L. and Q.Z.; visualization, J.L., H.L. and Q.Z.; supervision, Z.X.; project administration, Z.X. and Y.T.; funding acquisition, Y.T. and Z.X. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key Research and Development Program of China, grant number 2017YFC0405805 and the APC was funded by 2017YFC0405805.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets are available from the URLs provided in the article.

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

References

1. Guo, Q.; Zhuang, F.; Qin, C.; Zhu, H.; Xie, X.; Xiong, H.; He, Q. A Survey on Knowledge Graph-Based Recommender Systems. *IEEE Trans. Knowl. Data Eng.* 2020, Early Access. [CrossRef]
2. 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. [CrossRef]
3. Wang, X.; Wang, D.; Xu, C.; He, X.; Cao, Y.; Chua, T.-S. Explainable Reasoning over Knowledge Graphs for Recommendation. In Proceedings of the 33rd AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; pp. 5329–5336. [CrossRef]
4. Wang, Z.; Lin, G.; Tan, H.; Chen, Q.; Liu, X. CKAN: Collaborative Knowledge-aware Attentive Network for Recommender Systems. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, 25–30 July 2020; pp. 219–228. [CrossRef]
5. Wang, H.; Zhao, M.; Xie, X.; Li, W.; Guo, M. Knowledge Graph Convolutional Networks for Recommender Systems. In Proceedings of the 28th World Wide Web Conference, WWW 2019, San Francisco, CA, USA, 13–17 May 2019; pp. 3307–3313.. [CrossRef]
6. Wang, X.; Huang, T.; Wang, D.; Yuan, Y.; Liu, Z.; He, X.; Chua, T.-S. Learning Intents behind Interactions with Knowledge Graph for Recommendation. In Proceedings of the Web Conference 2021, Ljubljana, Slovenia, 19–23 April 2021; pp. 878–887. [CrossRef]
7. Wang, H.; Zhang, F.; Zhang, M.; Leskovec, J.; Zhao, M.; Li, W.; Wang, Z. Knowledge-aware Graph Neural Networks with Label Smoothness Regularization for Recommender Systems. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 968–977. [CrossRef]
8. 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. [CrossRef]
9. Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; Yu, P.S. A Comprehensive Survey on Graph Neural Networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 4–24. [CrossRef] [PubMed]
10. 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. [CrossRef]
11. Ai, Q.; Azizi, V.; Chen, X.; Zhang, Y. Learning Heterogeneous Knowledge Base Embeddings for Explainable Recommendation. *Algorithms* **2018**, *11*, 137. [CrossRef]
12. Wang, H.; Zhang, F.; Xie, X.; Guo, M. DKN: Deep Knowledge-Aware Network for News Recommendation. In Proceedings of the 2018 World Wide Web Conference on World Wide Web, Lyon, France, 23–27 April 2018; pp. 1835–1844. [CrossRef]
13. Goodfellow, I.J.; Bengio, Y.; Courville, A.C. Convolutional Networks. In *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016; pp. 330–372. Available online: http://www.deeplearningbook.org/contents/convnets.html (accessed on 14 December 2021).

14. 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 of the Asian Federation of Natural Language Processing, Beijing, China, 26–31 July 2015; pp. 687–696. [CrossRef]
15. Cao, Y.; Wang, X.; He, X.; Hu, Z.; Chua, T.-S. Unifying Knowledge Graph Learning and Recommendation: Towards a Better Understanding of User Preferences. In Proceedings of the 28th World Wide Web Conference, San Francisco, CA, USA, 13–17 May 2019; pp. 151–161. [CrossRef]
16. Hu, B.; Shi, C.; Zhao, W.X.; Yu, P.S. Leveraging Meta-path based Context for Top- N Recommendation with A Neural Co-Attention Model. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 1531–1540. [CrossRef]
17. Yu, X.; Ren, X.; Sun, Y.; Sturt, B.; Khandelwal, U.; Gu, Q.; Norick, B.; Han, J. Recommendation in heterogeneous information networks with implicit user feedback. In Proceedings of the 7th ACM Conference on Recommender Systems, Hong Kong, China, 12–16 October 2013; pp. 347–350. [CrossRef]
18. Zhao, H.; Yao, Q.; Li, J.; Song, Y.; Lee, D.L. Meta-Graph Based Recommendation Fusion over Heterogeneous Information Networks. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017; pp. 635–644. [CrossRef]
19. Luo, C.; Pang, W.; Wang, Z.; Lin, C. Hete-CF: Social-Based Collaborative Filtering Recommendation Using Heterogeneous Relations. In Proceedings of the 2014 IEEE International Conference on Data Mining, Shenzhen, China, 14–17 December 2014; pp. 917–922. [CrossRef]
20. Huang, X.; Fang, Q.; Qian, S.; Sang, J.; Li, Y.; Xu, C. Explainable Interaction-driven User Modeling over Knowledge Graph for Sequential Recommendation. In Proceedings of the 27th ACM International Conference on Multimedia, Nice, France, 21–25 October 2019; pp. 548–556. [CrossRef]
21. Sun, Z.; Yang, J.; Zhang, J.; Bozzon, A.; Huang, L.-K.; Xu, C. Recurrent knowledge graph embedding for effective recommendation. In Proceedings of the 12th ACM Conference on Recommender Systems, Vancouver, BC, Canada, 2 October 2018; pp. 297–305. [CrossRef]
22. Goodfellow, I.J.; Bengio, Y.; Courville, A.C. Sequence Modeling: Recurrent and Recursive Nets. In *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016; pp. 330–372. Available online: http://www.deeplearningbook.org/contents/rnn.html (accessed on 14 December 2021).
23. Wang, H.; Zhang, F.; Wang, J.; Zhao, M.; Li, W.; Xie, X.; Guo, M. RippleNet: Propagating User Preferences on the Knowledge Graph for Recommender Systems. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management, Torino, Italy, 22–26 October 2018; pp. 417–426. [CrossRef]
24. He, M.; Zhang, H.; Wen, H. RE-KGR: Relation-Enhanced Knowledge Graph Reasoning for Recommendation. In Proceedings of the Database Systems for Advanced Applications—26th International Conference, Taipei, Taiwan, 11–14 April 2021; pp. 297–305. [CrossRef]
25. Hamilton, W.L.; Ying, Z.; Leskovec, J. Inductive Representation Learning on Large Graphs. In Proceedings of the 31st Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 1024–1034. Available online: https://papers.nips.cc/paper/6703-inductive-representation-learning-on-large-graphs (accessed on 14 December 2021).
26. Schlichtkrull, M.S.; Kipf, T.N.; Bloem, P.; Berg, R.; Titov, I.; Welling, M. Modeling Relational Data with Graph Convolutional Networks. In Proceedings of the Semantic Web—15th International Conference, Heraklion, Crete, Greece, 3–7 June 2018; pp. 593–607. [CrossRef]
27. Niepert, M.; Ahmed, M.; Kutzkov, K. Learning Convolutional Neural Networks for Graphs. In Proceedings of the 33nd International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016; pp. 2014–2023. Available online: https://proceedings.mlr.press/v48/niepert16.html (accessed on 14 December 2021).
28. Mu, N.; Zha, D.; Gong, R. Gated Knowledge Graph Neural Networks for Top-N Recommendation System. In Proceedings of the 24th IEEE International Conference on Computer Supported Cooperative Work in Design, Dalian, China, 5–7 May 2021; pp. 1111–1116. [CrossRef]
29. Tu, K.; Cui, P.; Wang, D.; Zhang, Z.; Zhou, J.; Qi, Y.; Zhu, W. Conditional Graph Attention Networks for Distilling and Refining Knowledge Graphs in Recommendation. In Proceedings of the 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Gold Coast, QLD, Australia, 1–5 November 2021; pp. 1834–1843. [CrossRef]
30. Wang, Y.; Liu, Z.; Fan, Z.; Sun, L.; Yu, P.S. DSKReG: Differentiable Sampling on Knowledge Graph for Recommendation with Relational GNN. In Proceedings of the 30th ACM International Conference on Information and Knowledge Management, Gold Coast, QLD, Australia, 1–5 November 2021; pp. 3513–3517. [CrossRef]
31. Lin, Y.; Liu, Z.; Sun, M.; Liu, Y.; Zhu; X. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In Proceedings of the 29th AAAI Conference on Artificial Intelligence, Austin, TX, USA, 25–30 January 2015; pp. 2181–2187. Available online: https://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9571 (accessed on 14 December 2021).
32. Zhao, J.; Zhou, Z.; Guan, Z.; Zhao, W.; Ning, W.; Qiu, G.; He, X. IntentGC: A Scalable Graph Convolution Framework Fusing Heterogeneous Information for Recommendation. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 2347–2357. [CrossRef]

33. Huang, R.; Han, C.; Cui, L. Entity-aware Collaborative Relation Network with Knowledge Graph for Recommendation. In Proceedings of the 30th ACM International Conference on Information and Knowledge Management, Gold Coast, QLD, Australia, 1–5 November 2021; pp. 3098–3102. [CrossRef]
34. Lo, K.; Ishigaki, T. X-2ch: Quad-Channel Collaborative Graph Network over Knowledge-Embedded Edges. In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, 11–15 July 2021; pp. 2076–2080. [CrossRef]
35. Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; Bengio, Y. Graph Attention Networks. In Proceedings of 6th International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018. Available online: https://openreview.net/forum?id=rJXMpikCZ (accessed on 14 December 2021).
36. Hu, D. An Introductory Survey on Attention Mechanisms in NLP Problems. In Proceedings of the 2019 Intelligent Systems Conference, London, UK, 5–6 September 2019; pp. 432–448. [CrossRef]
37. Yang, Z.; Yang, D.; Dyer, C.; He, X.; Smola, A.J.; Hovy, E.H. Hierarchical Attention Networks for Document Classification. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego, CA, USA, 12–17 June 2016; pp. 1480–1489. [CrossRef]
38. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is All you Need. 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. 5998–6008. Available online: http://papers.nips.cc/paper/7181-attention-is-all-you-need (accessed on 14 December 2021).
39. The Last.FM Dataset. Available online: https://grouplens.org/datasets/hetrec-2011/ (accessed on 14 December 2021).
40. The Book-Crossing Dataset. Available online: https://grouplens.org/datasets/book-crossing/ (accessed on 14 December 2021).
41. The MovieLens 20M Dataset. Available online: https://grouplens.org/datasets/movielens/20m/ (accessed on 14 December 2021).
42. The Dianping-Food Dataset. Available online: https://www.dianping.com/ (accessed on 14 December 2021).
43. The Preprocessed Last.FM, Book-Crossing and MovieLens 20M Datasets and Their Corresponding Knowledge Graphs. Available online: https://github.com/weberrr/CKAN/tree/master/data (accessed on 14 December 2021).
44. The Preprocessed Dianping-Food Dataset and its Corresponding Knowledge Graph. Available online: https://github.com/hwwang55/KGNN-LS/tree/master/data/restaurant (accessed on 14 December 2021).
45. Rendle, S.; Freudenthaler, C.; Gantner, Z.; Schmidt-Thieme, L. BPR: Bayesian Personalized Ranking from Implicit Feedback. In Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, 18–21 June 2009; pp. 452–461.
46. Source Code of BPRMF, CKE and KGAT. Available online: https://github.com/xiangwang1223/knowledge_graph_attention_network/tree/master/Model (accessed on 14 December 2021).
47. Source Code of KPRN. Available online: https://github.com/xiangwang1223/KPRN (accessed on 14 December 2021).
48. Source Code of RippleNet. Available online: https://github.com/hwwang55/RippleNet (accessed on 14 December 2021).
49. Source Code of CKAN. Available online: https://github.com/weberrr/CKAN (accessed on 14 December 2021).
50. Source Code of KGCN. Available online: https://github.com/hwwang55/KGCN (accessed on 14 December 2021).
51. Source Code of KGNN-LS. Available online: https://github.com/hwwang55/KGNN-LS (accessed on 14 December 2021).
52. Source Code of KGIN. Available online: https://github.com/huangtinglin/Knowledge_Graph_based_intent_Network (accessed on 14 December 2021).
53. Aggarwal, C.C. Evaluating Recommender Systems. In *Recommender Systems*, 1st ed.; Springer: Cham, Switzerland, 2016; pp. 225–254. [CrossRef]
54. Herlocker, J.L.; Konstan, J.A.; Ried, J. An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Inf. Retr.* **2002**, *5*, 287–310. [CrossRef]
55. Hu, L.; Cao, L.; Wang, S.; Xu, G.; Cao, J.; Gu, Z. Diversifying Personalized Recommendation with User-session Context. In Proceedings of the 26th International Joint Conference on Artificial Intelligence, Melbourne, Australia, 19–25 August 2017; pp. 1858–1864. [CrossRef]