applied sciences

MDPI

Article

Knowledge Graph Extrapolation Network with Transductive Learning for Recommendation

Ruixin Ma, Fangqing Guo, Liang Zhao *, Biao Mei, Xiya Bu, Hao Wu and Enxin Song

School of Software Technology, Dalian University of Technology, Dalian 116024, China; maruixin@dlut.edu.cn (R.M.); guofangqing@mail.dlut.edu.cn (F.G.); aliezc0411@163.com (B.M.); buxiya1999@163.com (X.B.); wuhao990115@163.com (H.W.); 2019192405@mail.dlut.edu.cn (E.S.)
* Correspondence: liangzhao@dlut.edu.cn

Abstract: A knowledge graph is introduced into the personalized recommendation algorithm due to its strong ability to express structural information and exploit side information. However, there is a long tail phenomenon and data sparsity in real knowledge graphs, and most items are related to only a few triples. This results in a significant reduction in the amount of data available for training, and makes it difficult to make accurate recommendations. Motivated by these limitations, the Knowledge Graph Extrapolation Network with Transductive Learning for Recommendation (KGET) is proposed to improve recommendation quality. To be specific, the method first learns the embedding of users and items by knowledge propagation combined with collaborative signal to obtain high-order structural information, and the attention mechanism is used to distinguish the contributions of different neighbor nodes in propagation. In order to better solve with data sparsity and long tail phenomenon, transductive learning is designed to model links between unknown items to enrich feature representation to further extrapolate the knowledge graph. We conduct experiments with two datasets about music and books, the experiment results reveal that our proposed method outperforms state-of-the-art recommendation methods. KGET also achieves strong and stable performance in sparse data scenarios where items have merely a few triples.

Keywords: knowledge graph; recommendation; transductive learning; collaborative signal

Citation: Ma, R.; Guo, F.; Zhao, L.; Mei, B.; Bu, X.; Wu, H.; Song, E. Knowledge Graph Extrapolation Network with Transductive Learning for Recommendation. *Appl. Sci.* **2022**, *12*, 4899. https://doi.org/10.3390/app12104899

Academic Editor: Ángel González-Prieto

Received: 8 March 2022
Accepted: 9 May 2022
Published: 12 May 2022

1. Introduction

The personalized recommendation algorithm has a strong effect in assisting users to find items useful to them in a limited time. This method effectively alleviates the data explosion [1] problem brought on by the information age. Therefore, recommendation algorithms are widely used in many scenarios [2,3], such as online shopping, search engines, movie recommendation websites and so on.

Collaborative filtering (CF) [4,5], a traditional recommendation algorithm, has attracted much attention due to its effectiveness and universality. The main idea of CF is to predict the user's personal preference by mining the user's historical behavior information and then recommend similar products to users with similar preferences. However, the recommendation algorithm of collaborative filtering ignores side information, such as users' attributes, items' attributes and so on. Thus, the CF method suffers from data sparsity and cold start problem, where the user has a few interactions or no interactions. In order to exploit side information, some recommendation algorithms integrate deep learning methods [6,7] to obtain feature vectors of users and items, and calculate the probability of users' preference for items.

It is worth noting that users, items and their attribute information are related, instead of independent. Knowledge Graphs (KGs) are the multi-relational graphs which represent structured data. KGs consist of nodes and edges, where nodes represent entities and edges represent relations of entities. So, KGs are able to capture high-order structure information. Therefore, KGs [8–10] are introduced into the recommendation algorithm

because of their powerful ability to represent structured data. Wang et al. [11] proposed a deep knowledge-aware network, which combines knowledge graph representation and news recommendation and fuses semantic-level and knowledge-level representations of news. For the sequential recommendation, Huang et al. [12] proposed an explainable interaction-driven user modeling, recommended interpretability is provided by extracting paths between user-item pairs. Graph theory [13] is the study of graphs, which has been widely used in non-European graph data learning. Therefore, knowledge graph recommendations based on the graph neural network has become a popular solution, which utilizes embedding propagation of entities to aggregate higher-order neighborhood information. For example, Cao et al. [10] proposed a description-enhanced machine learning knowledge graph-based method, that combined the knowledge graph-based and text-based methods, which aggregates neighbors information by a graph neural network with attention.

In the real recommendation scenarios, there is a long tail effect [14], that is, a large amount of data has few interactions, and only a small portion of data has a large amount of interaction information. This is a challenge for recommendation because only a small amount of data can be used for training. Furthermore, the KG is dynamic, not static. The general methods based on KG are to make recommendations on the existing knowledge graph, and predict unknown entities through known entities, while ignoring the association between unknown entities, which also limits the accuracy of recommendations.

To address the limitations of existing methods, an end-to-end model, Knowledge Graph Extrapolation Network with Transductive Learning for Recommendation (KGET) is introduced. Specifically, KGET is equipped with two designs to tackle the challenges: (1) Embedding propagation learns the features of unknown entities by a knowledge propagation combined collaborative signal, and learns the weights of neighbor nodes to distill useful knowledge. It is able to capture collaborative and high-order structural information and long-distance interest. (2) Transductive learning is able to obtain more accurate and informative items representations, which models the relationships between unknown entities to enrich representations of items and realize extrapolate KG. It effectively alleviates the problem of inaccurate feature extraction due to too few available triples to entities.

Our main contributions are summarized as follows:

1. The novel recommendation method is proposed, Knowledge Graph Extrapolation Network with Transductive Learning for Recommendation (KGET), aiming to solve the long-tail problem and data sparsity with less triples information available to items in real recommendation scenarios.
2. The transductive learning strategy is designed to model relations between unknown entities to further propagate the knowledge, to obtain enrich representation for entities.
3. The experiments are conducted on two real-world datasets. The results show that the KGET significantly outperforms the baseline algorithm.

2. Related Work

Recommendations based on the knowledge graph have received more attention in recent years due to their excellent recommendation performance. In general, recommendations based on knowledge graphs can be categorized as three types: the embedding-based approach, path-based approach and hybrid approach. The embedding-based method [11,15,16] uses the knowledge graph as a feature extraction method, to make use of the rich semantic relationships in the knowledge graph according to some of the KG embedding models which embed entities and relationships in low dimensional space. (e.g., TransE [17], TransR [18]). The path-based approaches [19–21] extract association paths between the user and item, which leverages relationship information between entities and enhances the interpretability of recommendations. The hybrid approaches [9,22,23] combine semantic representation with path connectivity information, which use the multi-hop neighbors of items to enrich the representation information. However, to the best of our

knowledge, general current recommendation methods based on knowledge graphs mainly use a large number of training data sets for training, but items with a low number of interactions account for the majority of all items, which makes it difficult to recommend accurately. Our works focus on items with a few triples, and model relationships between unknown entities by relearning the feature of already-learned items via the transductive learning module, to enrich representation. Our proposed model is similar to the hybrid model.

3. Problem Formulation

In a recommendation scenario, the user set is represented by $\mathcal{U} = \{u_1, u_2, \ldots, u_M\}$ and the item set is represented by $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$. M is the number of users and N is the number of items. Graph theory is mainly used to study graphs. The graph is represented as $\mathcal{G} = \{V, E\}$, $V = \{v_1, \ldots, v_X\}$ is the set of nodes and $E = \{e_1, \ldots, e_Y\}$ is the set of edges. The knowledge graph is a data structure based on a graph, which is composed of nodes and edges. The node is the entity and the edge is the relationship between two entities, and the knowledge graph connects different kinds of heterogeneous information and can be regarded as a heterogeneous graph.

User-item Collaborative Graph (CG). It represents the user–item interaction, as well as the users' historical behavior data. $\mathcal{G}_1 = \{(u, y_{uv}, v) | u \in \mathcal{U}, v \in \mathcal{V}\}$ is denoted by the User-Item Collaborative Graph, where $y_{uv} = 1$ indicates that user u has interacted with item v; otherwise $y_{uv} = 0$. Thus, the User-Item Collaborative Graph is also a bipartite graph.

Knowledge Graph. The knowledge graph contains the side information in the recommendation, such as the attribute information of the item. Let $\mathcal{G}_2 = \{(h, r, t) \mid h \in \mathcal{E}, r \in \mathcal{R}, t \in \mathcal{E}\}$ denote the knowledge graph, where $\mathcal{E}$ is the set of entities, $\mathcal{R}$ is the set of relations, (h, r, t) is a set of triples. Herein, h denotes the head entity, r denotes the relation and t denotes the tail entity. For example, the triple (Ang Lee, Director of, Life of Pi) states the fact that Ang Lee is the director of the movie Life of Pi. The entities set consists of items and non-items. Thus, a set $\mathcal{A} = \{(v, e) | v \in \mathcal{V}, e \in \mathcal{E}\}$ is established to illustrate alignments between items in CG and entities in KG, where (v, e) indicates the item v is able to aligned with the entity e. The solid blue line illustrates the alignment process in Figure 1.

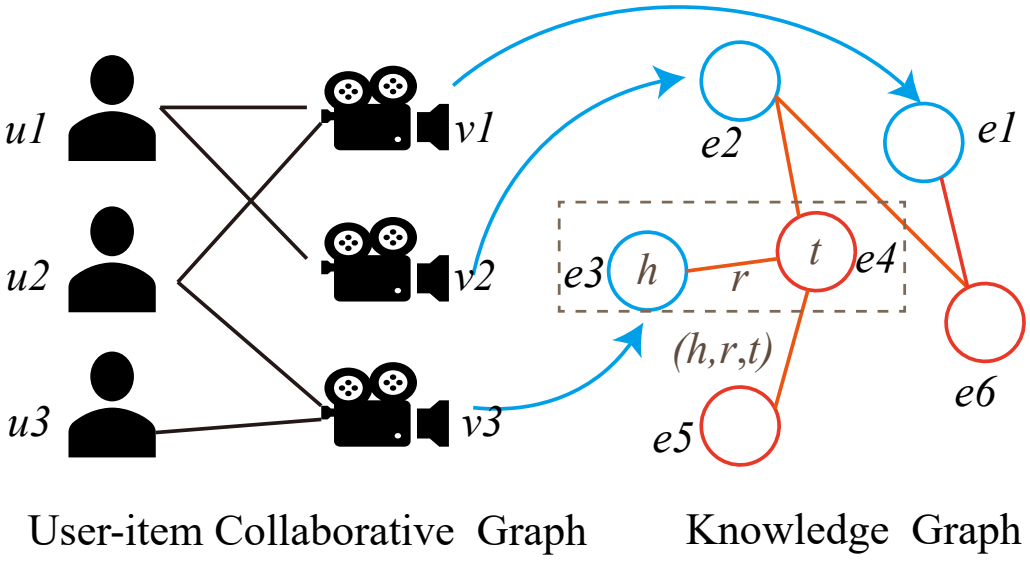

Figure 1. Illustration of Collaborative Graph and Knowledge Graph. The left is collaborative graph, which represents interaction information between users and items. The right is knowledge graph, where solid blue lines indicate the alignment of items in the collaborative graph with entities in the knowledge graph. The orange circles represent side information in recommendation, such as the attributes of the item.

Task Description. Given the user-item collaborative graph is $\mathcal{G}_1$ and knowledge graph is $\mathcal{G}_2$, the task is to predict the probability $\hat{y}_{uv}$ that user u would be fond of the item v.

The list of symbols in the model is shown in Table 1.

Table 1. List of symbols.

Symbol	Meaning
$\mathcal{U}$	Set of users
$\mathcal{V}$	Set of items
y_{uv}	Score between user and item
$\hat{y}_{uv}$	Predictive score between user and item
$\mathcal{E}$	Set of entities
$\mathcal{R}$	Set of relationships
$\mathcal{G}_1 = \{(u, y_{uv}, v) \| u \in \mathcal{U}, v \in \mathcal{V}\}$	User-item collaborative graph
$\mathcal{G}_2 = \{(h, r, t) \mid h \in \mathcal{E}, r \in \mathcal{R}, t \in \mathcal{E}\}$	Knowledge graph
$\mathcal{A}$	Set of alignments of items and entities
$\mathcal{I}_{(u)}$	Initial set of user
$\mathcal{I}_{(v)}$	Initial set of item
$\mathcal{N}^0_{(u)}$	Initial entity set of user
$\mathcal{N}^0_{(v)}$	Initial entity set of item
$\mathcal{N}^l_{(e)}$	Set of neighbors of entity at l-th order
$\mathcal{T}^l_{(e)}$	Triplet set of entity at l-th layer
$\mathbf{e}^l$	Embedding of entity at l-th layer
$\varphi = \{\mathbf{e}^{(1)}, \mathbf{e}^{(2)}, \ldots, \mathbf{e}^{(L)}\}$	Representations set of entity embedding in embedding propagation
φ_u	Representations set of user embedding in embedding propagation
φ_v	Representations set of item embedding in embedding propagation
ϕ_u	The final representations set of user embedding after transductive learning
ϕ_v	The final representations set of item embedding after transductive learning

4. The Proposed Method

In this section, the proposed KGET model is introduced in an end-to-end framework, which mainly consists of three components: (1) Embedding propagation, which learns the feature representation of each node in KG. (2) The transductive learning strategy, which models relationships between unknown entities themselves to enrich representations. (3) The prediction layer, which calculates the scores of users and corresponding items. The method is described in detail in the next few paragraphs. As shown in Figure 2, it illustrates the whole learning process.

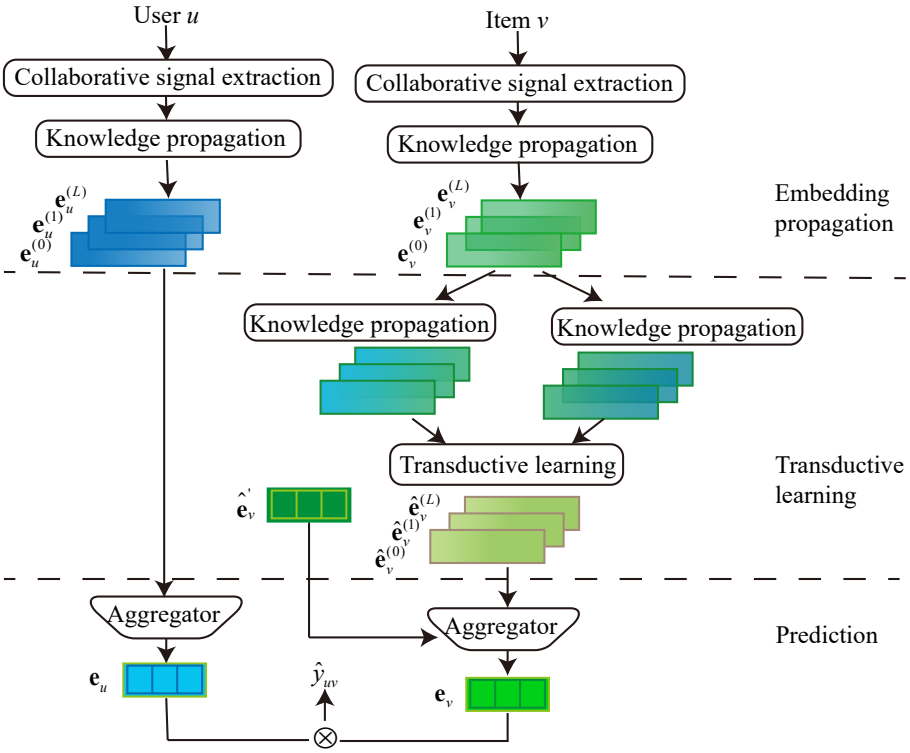

Figure 2. The illustration of the proposed KGET framework for recommendation.

4.1. Embedding Propagation

The embedding propagation is the knowledge propagation with collaborative signal. As is known to all, knowledge is propagated along the links in the knowledge graph. Thus, the user's long-distance interests are captured by capturing higher-order connection information in KG. Furthermore, the user-item collaborative graph with interactions between users and items, directly contains a collaborative signal, including the relationship between original user and item. So, the collaborative signal is extracted by learning interactions in CG. Concretely speaking, the embedding propagation is composed of two components: collaborative signal extraction and knowledge propagation. First, the collaborative signal is extracted as the initial sets of entities for users and items according to historical users' interaction data with the items. Second, knowledge is propagated layer by layer by aggregating information from each layer's neighborhood.

Collaborative signal extraction. In user and item interactions, it is generally assumed that users with similar access behaviors may prefer the similar items, and users may like items that are similar to their preferences. Thus, it makes sense for collaboration signals to be embedded in the feature representations of users and items and participate in propagation. In particular, the user's collaborative signal is represented by items, which the user has already indicated they prefer. A collection of these items with a score of 1 between user and items is named the initial set of users for the user. The initial set of users u formulated as $\mathcal{I}_{(u)} = \{v|y_{uv} = 1\}$. For the item, if different items are all associated with a user, there may be a correlation and similarity between these items. Therefore, first, users who have interacted with the item v are found, then other items that interact with the same users are treated as collaborative neighbors. The set of item's collaborative neighbors is called the initial set of items, as $\mathcal{I}_{(v)} = \{v_u|u \in \{u|y_{uv} = 1\} \wedge y_{uv_u} = 1\}$. For example, in Figure 1, user $u1$ prefers items $v1$ and $v2$, so the initial set of the user includes $v1$ and $v2$. For item $v1$, the user interacting with $v1$ is $u1$ and $u2$, $v2$ interacts with $u1$ and $v3$ interacts with $u2$. Thus, the initial set of the $v1$ contains $v2$ and $v3$.

Notice that the items in the user-item collaborative graph is aligned with the entities in the knowledge graph, to obtain the initial entity set of the user and the initial entity set of the item, respectively. The initial entity set of the user is shown below:

$$\mathcal{N}_{(u)}^{0} = \{e|(e,v) \in \mathcal{A} \wedge \ v \in \mathcal{I}_{(u)}\}. \tag{1}$$

The initial entity set of the item is shown below:

$$\mathcal{N}_{(v)}^{0} = \{e|(e,v_u) \in \mathcal{A} \wedge \ v_u \in \mathcal{I}_{(v)}\} \tag{2}$$

Knowledge propagation. The knowledge graph contains a large amount of side information, such as item attributes, correlation information and so on. In knowledge graph, entities are connected through relationships. The neighbor information of an entity has an essential influence on the feature of the entity. So, the feature representation of the entity is extended layer by layer through neighborhood knowledge along links between entities themselves. This method is able to capture high-order structural information.

Since the initial entity sets of users and items are subsets of the entity set $\mathcal{E}$ in the knowledge graph, the feature representations of users and items are extended and learned, which starts with them initial entity sets. In other words, both users and items carry out embedding propagation and feature extraction in the form of entities on the knowledge graph. Thus, the propagation of entity e is used to illustrate the specific propagation process for convenience.

Specifically, the entities directly connected to an entity e is the first-order neighbors of the entity, as $\mathcal{N}_{(e)}^{1}$. Furthermore, given a triple (h,r,t), the tail entity t is one of the first-order neighbors of the head entity h. By analogy, the l-order neighbors set of an entity is defined as $\mathcal{N}_{(e)}^{l}$ at layer l. The details as follows:

$$\mathcal{N}_{(e)}^{l} = \{t|(h,r,t) \in \mathcal{G}_2 \wedge \ h \in \mathcal{N}_{(e)}^{l-1}\}, \tag{3}$$

where $\mathcal{N}_{(e)}^{l-1}$ denotes the $(l-1)$-order neighbors of the entity e at layer $l-1$. The triplet set composed of all neighbors at layer l for entity e is shown below:

$$\mathcal{T}_{(e)}^{l} = \{(h,r,t)|(h,r,t) \in \mathcal{G}_2 \wedge \ h \in \mathcal{N}_{(e)}^{l-1}\}. \tag{4}$$

In the real scene, different tail entities have different contributions to feature representation of an entity under different conditions. For example, user u_1 prefers a movie named Crouching Tiger, Hidden Dragon. The movie is both a martial arts film and a love story. However, user u_1 actually prefers this movie because he likes martial arts films. Therefore, martial arts as the attribute of the film should have a higher weight than love story. Therefore, the attention mechanism is designed to learn the weights of different tail entities. The global embedding of the triple set at the l-th layer for the entity is defined as follows:

$$\mathbf{e}^{l} = \sum_{(h,r,t) \in \mathcal{T}_{(e)}^{l}} \pi(h,r)\mathbf{e}^{t}, \tag{5}$$

where $\mathbf{e}^{t}$ is the embedding representation of the tail entity t, $\pi(h,r)$ is a scoring function used to compute the attention weight, which controls how much information from the tail entity is propagated to head entity in this case, where head entity is h and the relation is r.

The $\pi(h,r)$ is implemented via the attention network, which is formulated as follows:

$$\pi(h,r) = \sigma(\mathbf{W}_2 \mathrm{ReLU}(\mathbf{W}_1 \mathbf{z}_0 + \mathbf{b}_1) + \mathbf{b}_2), \tag{6}$$

$$\mathbf{z}_0 = \mathrm{ReLU}\Big(\mathbf{W}_0\Big(\mathbf{e}^{h} \| \mathbf{r}\Big) + \mathbf{b}_0\Big), \tag{7}$$

where $\pi(\cdot)$ is a multi-layer neural network. $\mathbf{W}_0$, $\mathbf{W}_1$ and $\mathbf{W}_2$ are the trainable weight matrices of each layer, respectively. $\mathbf{b}_0$, $\mathbf{b}_1$ and $\mathbf{b}_2$ are the biases. $\sigma(\cdot)$ is the last nonlinear activation function, which is Sigmoid. ReLU is the activation function except for the last layer. Then, the softmax function is applied to normalize the coefficients across all triples, as follows:

$$\pi(h,r) = \frac{\exp(\pi(h,r))}{\sum_{(h',r',t')\in\mathcal{T}_e^l}\exp(\pi(h',r'))}. \tag{8}$$

The attention score function is capable of attracting more attention to the neighboring tail entity that has more influence. The method distills more accurate association information.

Finally, the representations of entities of other layers are computed according to Equation (5), and the representations set from layer 1 to layer L are obtained for the entity e:

$$\varphi = \{\mathbf{e}^{(1)}, \mathbf{e}^{(2)}, \ldots, \mathbf{e}^{(L)}\}, \tag{9}$$

So, the representation sets of user and item are obtained as φ_u and φ_v. However, there is still an issue to note, which is that the importance of the initial entity set should not be ignored. In particular, the initial entity set of the user is directly related to the user and represents the user's preferred items. It directly reflects user interest and, for items, the initial entity set is a set of collaborative neighborhoods, which contains collaboration information and strong interaction information. Thus, the initial entity sets of users and items are treated as their 0-order neighborhood, respectively. The average value of the embedding representations of the entities in the initial entity set of the user and the item is regarded as the 0-th layer embedding of the user and the item:

$$\mathbf{e}^{(0)} = \frac{\sum_{\mathbf{e}\in\mathcal{N}_{(e)}^0}\mathbf{e}}{\left|\mathcal{N}_{(e)}^0\right|}, \tag{10}$$

Finally, the representation sets of users and items, which contain attention weight, are as follows:

$$\varphi_u = \{\mathbf{e}_u^{(0)}, \mathbf{e}_u^{(1)}, \ldots, \mathbf{e}_u^{(L)}\}, \tag{11}$$

$$\varphi_v = \{\mathbf{e}_v^{(0)}, \mathbf{e}_v^{(1)}, \ldots, \mathbf{e}_v^{(L)}\}. \tag{12}$$

4.2. Transductive Learning

The long tail problem is a serious challenge for personalized recommendation. In the real recommended scenario, most entities have only a few interactive data. This results in insufficient data available for training. However, the general methods require a large amount of training data to achieve an accurate effect. Therefore, the transductive learning strategy is proposed to to alleviate the long tail problem. Transductive learning is applied to each layer of item individually. For ease of understanding, we use the example of executing the strategy at layer l, as well as at other layers. Here are the details.

First, the representations of items are updated, based on the representation of unknown items already learned in the previous step. Then, considering the small number of triples available to entities and high uncertainties on unknown items, the novel representations of items are fed into two individual knowledge propagation layers to model relations of unknown items. So, the number of nodes and layer depth of knowledge propagation in transductive learning are consistent with the number of nodes and layer depth in embedding propagation. Then, the transductive learning layer is designed to re parameterize the model by computing the output representations across the two knowledge propagation layers. The specific calculation process is as follows:

$$\hat{\mathbf{e}}_v^l = \sum_{(h,r,t)\in\mathcal{T}_{(e)}^l}\pi(h,r)\mathbf{e}^t + \exp\{0.5\cdot[\sum_{(h,r,t)\in\mathcal{T}_{(e)}^l}\pi(h,r)\mathbf{e}^t]'\}, \tag{13}$$

where the learning feature embedding of items that have been updated in the previous step is φ_v , and φ_v^l denotes representation of item v at l-th layer. $\sum_{(h,r,t)\in\mathcal{T}_{(e)}^l} \pi(h,r)\mathbf{e}^t$ and $[\sum_{(h,r,t)\in\mathcal{T}_{(e)}^l} \pi(h,r)\mathbf{e}^t]'$ represent processes through two individual knowledge propagation layers, respectively. Meanwhile, $\mathbf{e}^h \in \varphi_v^l$, if there is an item v and entity h that satisfy $(v,h) \in \mathcal{A}$. Homoplastically, $\mathbf{e}^t \in \varphi_v^l$, if there is an item v and entity t that satisfy $(v,t) \in \mathcal{A}(e,t) \in \mathcal{A}$. $\hat{\mathbf{e}}_v^l$ is defined as embedding of item v at the l-th layer, which contains a relationship between unknown items that have been modeled.

Transductive learning predicts the relationship between unknown entities themselves and constructs a connection for them. As a result, when a new item is added to the knowledge graph, reliable contact information for that entity is also capable of being obtained, even though the new entity has no triple. This approach extends the knowledge graph and also eases the cold starts problem.

Furthermore, different from users who only appear in the user-item collaborative graph, items appear both in the collaborative graph and as entities in the knowledge graph. This results in the item having initial embedding in the knowledge graph, and the embedding of the item in the original knowledge graph $\hat{\mathbf{e}}_v^{'}$ is closely related to the item, so the initial embedding of the item is added to the representation set.

Performing the above procedure at each layer results in the new embedding representations of the item at each layer with richer connection information. Then, the representations of items is updated again to $\phi_v = \{\hat{\mathbf{e}}_v^{'}, \hat{\mathbf{e}}_v^{(0)}, \hat{\mathbf{e}}_v^{(1)}, \ldots, \hat{\mathbf{e}}_v^{(L)}\}$. The representations of the user is also φ_u. For convenience, let us say ϕ_u is equal to φ_u. Therefore, the final embedding of the user and item is shown below:

$$\phi_v = \{\hat{\mathbf{e}}_v^{'}, \hat{\mathbf{e}}_v^{(0)}, \hat{\mathbf{e}}_v^{(1)}, \ldots, \hat{\mathbf{e}}_v^{(L)}\} \tag{14}$$

$$\phi_u = \{\mathbf{e}_u^{(0)}, \mathbf{e}_u^{(1)}, \ldots, \mathbf{e}_u^{(L)}\} \tag{15}$$

4.3. Prediction Layer

The embedding representation of each layer contains the underlying latent feature of the item and the user, as well as higher-order connectivity information. It is useful for mining the user's underlying preferences. Hence, it is indispensable to aggregate information of each layer of user and item representation. There are three types of aggregators:

- Sum aggregator, which sums embedding representations of each layer up before applying a non-linear transformation, as follows:

$$f_{\text{sum}} = \sigma(\mathbf{W} \cdot \sum\nolimits_{\mathbf{e}\in\phi} \mathbf{e} + \mathbf{b}), \tag{16}$$

  where $\mathbf{W}$ and $\mathbf{b}$ are the trainable weight and bias, σ is set as an activation function as Sigmoid. ϕ denotes the representation of users or items at all layers. If the representations of all layers of the item need to be combined, ϕ represents ϕ_v and $\mathbf{e}$ represents each of these representations in ϕ_v. The same goes for the user. The other two aggregators are also represented this way.
- Pool aggregator, which selects the maximum vector in the feature representations of all layers and uses a non-linear transformation as follows:

$$f_{\text{pool}} = \sigma(\mathbf{W} \cdot \text{pool}_{\max}(\phi) + \mathbf{b}). \tag{17}$$

- Concat aggregator, which concatenates representation vectors followed by a non-linear function as follows:

$$f_{\text{concat}} = \sigma\Big(\mathbf{W} \cdot \Big(\mathbf{e}^{(1)} \| \ldots \| \mathbf{e}^{(k)} \| \ldots \| \mathbf{e}^{(|\phi|)}\Big) + \mathbf{b}\Big), \tag{18}$$

where $|\phi|$ is the number of vectors in ϕ, $\mathbf{e}^{(k)} \in \phi$, $\|$ is the concatenation operation which means concatenate two vectors.

The use of aggregators which combines the representation of different layers is able to capture higher-order structural information in the knowledge graph and enrich the potential representation of users and items. The effects of different aggregators are compared in Section 5.5.3.

The aggregated vectors of the user and item were obtained in the previous step. Let $\mathbf{e}_u$ denote the vector representation of the user. $\mathbf{e}_v$ is set as the vector representation of the item. Finally, the inner product of the user representation and item representation is conducted to predict the probability of user preference for the item:

$$\hat{y}_{uv} = \mathbf{e}_u{}^\top \mathbf{e}_v. \tag{19}$$

4.4. Loss Function

Note that the negative sample is just as important as the positive sample, so the cross-entropy loss is opted to optimize the recommendation model:

$$\mathcal{L} = \sum_{u \in \mathcal{U}} \left(\sum_{v \in \{v|(u,v) \in \mathcal{Z}^+\}} \mathcal{J}(y_{uv}, \hat{y}_{uv}) - \sum_{v \in \{v|(u,v) \in \mathcal{Z}^-\}} \mathcal{J}(y_{uv}, \hat{y}_{uv}) \right) + \gamma \|\Theta\|_2^2, \tag{20}$$

where $\mathcal{J}$ is the cross-entropy loss, $\mathcal{Z}^+$ indicates the positive interactions between users and items, while $\mathcal{Z}^-$ is the sampled negative interactions set, which consists of user-item pairs between users and items they do not like and score between the user and item is 0, $\|\Theta\|_2^2$ is the L_2 regularization parameterized by γ.

5. Experiments

The proposed model KGET is evaluated on the real-world datasets in this part. The following research questions will be answered:

- RQ1: How does our proposed KGET perform compared to the baseline algorithm based on the knowledge graph?
- RQ2: How does choosing different parameters affect KGET?

5.1. Dataset

To evaluate the performance of KGET, we conducted the experiment in two scenarios: music recommendations and book recommendations. So, we used two benchmark datasets: Last.FM and Book-Crossing, respectively. These datasets were obtained from different domains and have different sizes and sparsity.

Last.FM. The dataset is provided by last.fm online music system, which contains music artist listening information from 2 thousand users, where tracks are defined as items.

Book-Crossing. The dataset contains readers' ratings of different books from the book-crossing community. The ratings range from 0 to 10.

Because these two datasets with explicit ratings are not suitable to be applied to the proposed model KGET, the explicit ratings are transformed explicit feedbacks. Thus, 1 indicates positive samples that the user has rated the items positively. On the contrary, 0 is marked as negative samples which are randomly sampled from unrated items for the user. Due to the sparsity of Last.FM and Book-Crossing, no threshold is set, and all ratings are viewed as the positive samples. Besides the interactions, the knowledge graph about the item needed to be constructed for the two datasets. We followed the work of [24] to construct sub-KGs for Last.FM and Book-Crossing from Microsoft Satori KG. The confidence level of the subset is greater than 0.9.

To verify the proposed method KGET in data sparsity, where items have few triples, there are still a large number of items with more triples removed from the current dataset. The current dataset was processed as follows: First, we measured the frequency of items

appearing in the knowledge graph. Second, we extracted random entities that appear less frequently. Finally, we only retained user interaction information associated with the extracted entities. (1) Last.FM contains 9366 entities and 60 relations which is used for the recommendation task. Particularly, we sample the items which have associated triplets between 3 and 40 in KG. Finally, there are 13,729 user–item interactions information extracted from 42,346 interactions. (2) Book-Crossing consists of 77,903 entities and 25 relations. We randomly sampled the items which have associated triplets between 4 and 80 in KG, and there are 11,615 user-item interactions information extracted from 139,746 interactions.

We selected 60% of user–item interactions as the training set, 20% of the remaining interactions data were used as the validation set, and the rest of the interactions serve as the test set. This method ensures that the three subsets are not duplicated. Training sets were used to train the model. We fine-tuned the parameters in the validation set. We evaluated the performance of the model in the test set.

The detail statistics for two datasets are shown in Table 2.

Table 2. Basic statitics for datasets.

	Last.FM	**Book-Crossing**
#user	1863	5275
#item	1024	957
#interactions	13,729	11,615
#entities	9366	77,903
#relations	60	25
#KG triples	15,518	151,500

5.2. Baselines

We compare the proposed model with the following baselines to demonstrate the effectiveness of KGET. The baselines are divided into two types: KG-free method and KG-aware methods. The KG-free method is a CF-based method (BPRMF), and KG-aware methods include embedding-based (CKE), path-based (RippleNet) and hybrid-based (KGCN, KGNN, KGAT, CKAN).

BPRMF [25]. The method belongs to a CF model, which uses matrix factorization for item recommendation from implicit feedback, and directly optimizes for ranking by proposed Bayesian personalized ranking from implicit feedback.

CKE [16]. The model is an embedding-based method, which learns the latent representations in collaborative filtering, and extracts items' semantic representations with structural, textual and visual information by leveraging the heterogeneous information in the knowledge graph.

RippleNet [21]. The model is a path-based method with a knowledge embedding-based method, which thinks of the preference propagation as ripples in water, and automatically discovers the path between users and items that they might prefer.

KGCN [24]. The method is a non-spectral GCN approach in the knowledge graph, which extends the receptive field to multiple hops for capturing high-order structural information and users' preference, and user-relation scores are calculated as neighbor weights.

KGNN [26]. The method proposes knowledge-aware graph neural networks with label smoothness regularization, which transforms a heterogeneous KG into a user-personalized weighted graph that characterizes user's preferences, and then uses a supervised fashion to train the edge weights. To avoid overfitting, the technique for regularization of edge weights is designed to generalize to unobserved interactions.

KGAT [27]. The method combines a user-item collaboration graph and knowledge graph into a collaborative knowledge graph, which explicitly models the high-order relations. The knowledge-aware attention mechanism is adopted to learn the weights of neighbor nodes.

CKAN [23]. The method applies a heterogeneous propagation strategy to explicitly encode collaborative signals and knowledge associations, and a knowledge-aware attention mechanism is employed to discriminate the importance of neighbors.

5.3. Experimental Settings

We evaluate the proposed method KGET in click-through rate (CTR), which predicts the matching score between user and item. The AUC and F1 are used as the metrics to evaluate the model in the test set. AUC is the area under the curve which is defined as the area enclosed under the ROC (Receiver Operating Characteristic) curve. F1 is the harmonic average of precision and recall. The higher the value of AUC and F1, the better the performance of the model.

The proposed method KGET is implemented by PyTorch, and the optimizer is Adam [28] for the model. We apply a grid search for hyper-parameters. We set batch size as 2048. The learning rate is selected from $\left\{10^{-3}, 2 \times 10^{-3}, 10^{-2}, 2 \times 10^{-2}\right\}$. The coefficient of $L2$ normalization is tuned amongst $\left\{2 \times 10^{-6}, 10^{-5}, 2 \times 10^{-4}, 10^{-3}\right\}$. The depth of layers is set to $\{1, 2, 3, 4\}$. The embedding size is set to $\{4, 8, 16, 32, 64, 128, 256\}$. The triple set size is searched in $\{4, 8, 16, 32, 64, 128\}$. Baseline algorithms are parameterized according to the best parameter settings in their original papers and experimental verification.

5.4. Prediction Performance (RQ1)

In the section, we first focus on the performance of all methods. Table 3 presents the performance comparison results with the AUC and F1 of eight methods. The observations are as follows:

- The proposed method KGET consistently outperforms these state-of-the-art baselines on all datasets. KGET exhibits improvements of 1.6% and 4.5% over the best algorithms in AUC on Last.FM and Book-Crossing, respectively, and F1 increased by 1.2% and 1.4% in Last.FM and Book-Crossing, respectively. The improvements of our method on Book-Crossing is higher than results on Last.FM. This may be due to the fact that Book-Crossing is sparser than Last.FM, and the distribution of users and items are more unbalanced in Book-Crossing, with the number of users far outnumbering the number of items. This demonstrates that direct learning of embedding unseen entities is difficult to capture comprehensive relationship information. However, our proposed method KGET is able to learn embedding more stably in training data sparsity and data imperfection by learning the connections between unseen entities (transductive learning).
- The performance of RippleNet, KGCN and KGNN in Last.FM is better than the performance in Book-Crossing, which indicates that the user's neighbor information is also useful in recommendations. Since RippleNet, KGCN and KGNN do not focus on the user's neighborhood information, but only on the neighborhood of the item with which the user interacts, and there are many more users than items in Book-Crossing, the user's neighborhood information becomes more important.
- We observe that CKAN and KGAT achieve better performance than other baselines, which illustrates that a method combining the collaborative signal with side information in KG might fully mine the feature information of users and items. The collaborative signal and higher-order connection information benefit the personalized recommendation.
- Compared with RippleNet, the results show that the performance of propagation-based methods outperform the path-based methods. Which may be because the relationships between entities in the real knowledge graph are complicated and it is difficult to extract high-quality paths.
- The AUC of the KG-free method BPRMF is better than that of KG-based methods including CKE, RippleNet, KGCN, and KGNN, indicating the advantages of the collaborative filtering method. It is not certain that good results are obtained by using

side information of the knowledge graph. It is also necessary to consider whether the advantages of the knowledge graph might be fully utilized and whether it is suitable for the dataset.

Table 3. Overall performance comparison.

Model	Last.FM		Book-Crossing	
	Auc	F1	Auc	F1
BPRMF	0.773 (−9.0%)	0.683 (−16.1%)	0.659 (−15.4%)	0.605 (−22.8%)
CKE	0.763 (−10.1%)	0.669 (−17.8%)	0.673 (−13.6%)	0.600 (−23.5%)
RippleNet	0.693 (−18.4%)	0.685 (−15.8%)	0.544 (−30.2%)	0.663 (−15.4%)
KGCN	0.704 (−17.1%)	0.707 (−13.1%)	0.521 (−33.1%)	0.743 (−5.2%)
KGNN	0.669 (−21.2%)	0.701 (−13.9%)	0.525 (−32.6%)	0.746 (−4.8%)
CKAN	0.827 (−2.6%)	0.804 (−1.2%)	0.734 (−5.8%)	0.773 (−1.4%)
KGAT	0.835 (−1.6%)	0.773 (−5.0%)	0.744 (−4.5%)	0.720 (−8.2%)
Ours	0.849	0.814	0.779	0.784

5.5. Study of KGET (RQ2)

In this section, we investigate the performance of the proposed model KGET under different hyper-parameter conditions, which include the dimension of embedding, triple set size, aggregators, transductive learning and depth of layer.

5.5.1. Effect of Dimension of Embedding

We show the results of different dimensions of embedding on all datasets on Figures 3–6. We can observe that AUC and F1 values gradually increase with the increase in dimension of embedding. However, when the dimension exceeds the threshold, the related indicators decrease, especially in Book-Crossing. This may be because the embedding vector encodes more feature information as the dimension increases, but when the dimension is too large, it will cause overfitting.

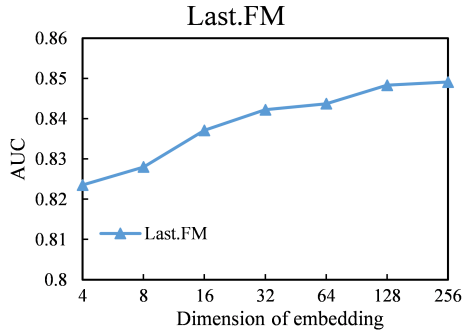

Figure 3. Comparison of the AUC performance on different embedding dimensions in Last.FM.

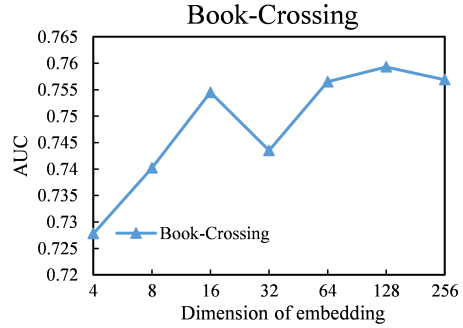

Figure 4. Comparison of the AUC performance on different embedding dimensions in Book-Crossing.

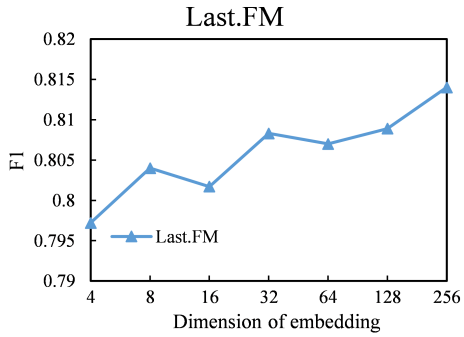

Figure 5. Comparison of the F1 performance on different embedding dimensions in Last.FM.

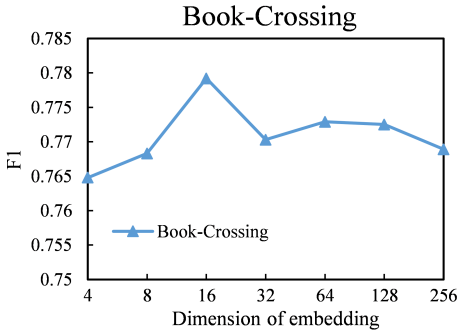

Figure 6. Comparison of the F1 performance on different embedding dimensions in Book-Crossing.

5.5.2. Effect of Triple Set Size

As shown in Tables 4–7, we set the user and item triplet sizes for different sizes, and then observe the values of AUC and F1 on Last.FM and Book-Crossing. We conducted the experiment with all parameters remaining consistent except the size of the triplet. We set the depth of layer as 1, weight of $L2$ regularization term as 10^{-5}, learning rate as 2×10^{-3}, dimension of entity and relation embedding as 128, and the sum aggregator was used uniformly.We found that the KGET achieves the best AUC performance when the size of the user's triple set is 4 and the number of triples for the item is set to 16 on Last.FM, and the best value of F1 be obtained when number of triples for user is set to 4 and the number of triples for item is set to 64. On Book-Crossing, we can observe that the AUC value reaches the best when sampling 32 triples for user and 4 triples for item, and the best F1 result appears when sampling 64 triples for user and 8 triples for item.

Table 4. The effect of size of triple set on AUC in Last.FM.

Item \ User	4	8	16	32	64	128
4	0.828	0.832	0.827	0.836	0.835	0.830
8	0.837	0.840	0.837	0.838	0.832	0.832
16	0.848	0.841	0.834	0.835	0.837	0.835
32	0.842	0.837	0.838	0.833	0.833	0.834
64	0.839	0.829	0.825	0.827	0.824	0.830
128	0.831	0.830	0.816	0.821	0.818	0.820

Table 5. The effect of size of triple set on F1 in Last.FM.

Item \ User	4	8	16	32	64	128
4	0.788	0.803	0.797	0.805	0.805	0.804
8	0.798	0.808	0.805	0.801	0.804	0.802
16	0.809	0.805	0.804	0.793	0.805	0.805
32	0.807	0.808	0.808	0.804	0.804	0.804
64	0.813	0.81	0.804	0.809	0.805	0.804
128	0.811	0.803	0.804	0.803	0.804	0.807

Table 6. The effect of size of triple set on AUC in Book-Crossing.

Item \ User	4	8	16	32	64	128
4	0.741	0.742	0.736	0.759	0.746	0.749
8	0.739	0.743	0.737	0.735	0.750	0.747
16	0.750	0.737	0.726	0.737	0.742	0.739
32	0.731	0.733	0.717	0.73	0.726	0.731
64	0.722	0.718	0.717	0.713	0.720	0.712
128	0.722	0.717	0.706	0.703	0.710	0.719

Table 7. The effect of size of triple set on F1 in Book-Crossing.

Item \ User	4	8	16	32	64	128
4	0.771	0.778	0.764	0.772	0.771	0.769
8	0.771	0.772	0.776	0.771	0.782	0.771
16	0.778	0.776	0.773	0.772	0.777	0.772
32	0.772	0.774	0.774	0.772	0.772	0.770
64	0.769	0.778	0.777	0.773	0.777	0.772
128	0.776	0.778	0.777	0.772	0.773	0.775

Generally speaking, the larger the sample of the set of triples, the more neighborhood information is propagated. However, a larger size of triple set might also cause data redundancy and propagate more noise data, which cause the performance of the model to degrade. Furthermore, the model achieves the best performance when there are more item's triples than user's triples on Last.FM, and conversely, the model's performance is best when there are more user's triples than item's triples on Book-Crossing. A reasonable explanation is that the number of users in the book dataset is more than the number of items, while the number of items in the music dataset is more than the number of items, so the model learns feature information by enlarging the user's triple set in the book dataset and vice versa.

5.5.3. Effect of Aggregators

We experimented with three types of aggregators: sum aggregator (KGET-sum), concat aggregator (KGET-concat) and pool aggregator (KGET-pool), the detailed results are shown in Tables 8 and 9. In most cases, the sum aggregator works better than the concat aggregator and pool aggregator. The possible reason is that the method of summing each layer of vectors retains more information about the hidden layer.

Table 8. The influence of aggregator in AUC.

Aggregator	KGET-Concat	KGET-Pool	KGET-Sum
Last.FM	0.824	0.824	0.849
Book-Crossing	0.731	0.744	0.759

Table 9. The influence of aggregator in F1.

Aggregator	KGET-Concat	KGET-Pool	KGET-Sum
Last.FM	0.788	0.799	0.813
Book-Crossing	0.772	0.780	0.773

5.5.4. Effect of Transductive Learning

To order to explore the effect of transductive learning, we carried out the experiment without using the transductive learning strategy. The results of AUC and F1 on datasets are presented in Table 10. The KGETNO denotes the method without transductive learning. We found the method without transductive learning performs worse than the KGET, which verifies the necessity of the existence of transductive learning. It proves that the prediction of relationships between unseen entities extrapolates the knowledge graph and is beneficial for recommending tasks.

Table 10. The effect of transductive learning.

	Last.FM		Book-Crossing	
	KGET	KGETNO	KGET	KGETNO
AUC	0.848	0.821	0.778	0.732
F1	0.814	0.793	0.776	0.767

5.5.5. Influence of Depth of Layer

The depth of layer was changed from 1 to 4, to study the effect of depth of layer. Tables 11 and 12 summarize the performance of the model at different depths of layer. We observed that the KGET works best with a depth of 1 in Last.FM and achieves the best performance at a depth of 2 in Book-Crossing, which if too deep may bring the over-smoothing problem and have a negative impact on model performance in the graph neural network, especially in the case of large amounts of data.

Table 11. The Influence of depth of layer on Last.FM.

Layer	1	2	3	4
AUC	0.849	0.835	0.832	0.818
F1	0.814	0.795	0.802	0.792

Table 12. The Influence of depth of layer on Book-Crossing.

Layer	1	2	3	4
AUC	0.759	0.773	0.755	0.748
F1	0.773	0.784	0.776	0.766

6. Conclusions

In this paper, we propose a novel end-to-end model, Knowledge Graph Extrapolation Network with Transductive Learning for Recommendation, which is named KGET. KGET learns the embedding of users and items by embedding propagation to obtain high-order connection information in KG, which is composed of knowledge propagation combined with collaborative signal, and the attention mechanism is designed to distinguish contributions of different neighbor nodes. In this study, transductive learning was employed to model links between unknown entities and enrich representation of items, which re parameterize by representations learned via two individual knowledge propagation layers. The strategy effectively alleviated data sparsity and cold start problems because of the full use of inter-entity relationship information. Extensive experiments have shown that the proposed model KGET outperformed the relevant baselines.

This work was mainly to enrich the feature representations of items by learning the relationships between items. In future work, we plan to mine more relationship information in knowledge graph and focus on fine-grained interactions about users and items, to further improve our model. We will conduct more experiments to demonstrate the validity of our model.

Author Contributions: Conceptualization, F.G. and B.M.; Data curation, X.B.; Formal analysis, X.B.; Funding acquisition, L.Z.; Methodology, F.G.; Project administration, R.M.; Software, B.M. and H.W.; Supervision, R.M.; Validation, H.W.; Writing—original draft, F.G. and E.S.; Writing—review and editing, F.G. and L.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the National Natural Science Foundation of China (61906030), the Youth Science and Technology Star Support Program of Dalian, 2021RQ057 and the Natural Science Foundation of Liaoning Province (2020-BS-063).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

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

References

1. Ying, R.; He, R.; Chen, K.; Eksombatchai, P.; Hamilton, W.L.; Leskovec, J. Graph convolutional neural networks for web-scale recommender systems. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 974–983.
2. Jin, J.; Qin, J.; Fang, Y.; Du, K.; Zhang, W.; Yu, Y.; Zhang, Z.; Smola, A.J. An efficient neighborhood-based interaction model for recommendation on heterogeneous graph. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Online, 6–10 July 2020; pp. 75–84.
3. Gong, J.; Wang, S.; Wang, J.; Feng, W.; Peng, H.; Tang, J.; Yu, P.S. Attentional graph convolutional networks for knowledge concept recommendation in moocs in a heterogeneous view. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, Xi'an, China, 25–30 July 2020; pp. 79–88.
4. Xue, F.; He, X.; Wang, X.; Xu, J.; Liu, K.; Hong, R. Deep Item-based Collaborative Filtering for Top-N Recommendation. *ACM Trans. Inf. Syst.* **2019**, *37*, 33:1–33:25. [CrossRef]
5. Wang, H.; Wang, J.; Zhao, M.; Cao, J.; Guo, M. Joint topic-semantic-aware social recommendation for online voting. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, Singapore, 6–10 November 2017; pp. 347–356.
6. Chen, C.; Zhang, M.; Ma, W.; Liu, Y.; Ma, S. Efficient Non-Sampling Factorization Machines for Optimal Context-Aware Recommendation. In Proceedings of the WWW '20: The Web Conference 2020, Taipei, Taiwan, 20–24 April 2020; pp. 2400–2410.
7. Liu, J.; Zhao, P.; Zhuang, F.; Liu, Y.; Sheng, V.S.; Xu, J.; Zhou, X.; Xiong, H. Exploiting Aesthetic Preference in Deep Cross Networks for Cross-domain Recommendation. In Proceedings of the WWW '20: The Web Conference 2020, Taipei, Taiwan, 20–24 April 2020; pp. 2768–2774.
8. Cao, Y.; Wang, X.; He, X.; Hu, Z.; Chua, T. Unifying Knowledge Graph Learning and Recommendation: Towards a Better Understanding of User Preferences. In Proceedings of the World Wide Web Conference, WWW 2019, San Francisco, CA, USA, 13–17 May 2019; pp. 151–161.
9. 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, Online, 19–23 April 2021; pp. 878–887.
10. Cao, X.; Shi, Y.; Yu, H.; Wang, J.; Wang, X.; Yan, Z.; Chen, Z. DEKR: Description Enhanced Knowledge Graph for Machine Learning Method Recommendation. In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Online, 11–15 July 2021; pp. 203–212.
11. Wang, H.; Zhang, F.; Xie, X.; Guo, M. DKN: Deep knowledge-aware network for news recommendation. In Proceedings of the 2018 World Wide Web Conference, Lyon, France, 23–27 April 2018; pp. 1835–1844.
12. 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.
13. Irfan, M.; Rehman, H.U.; Almusawa, H.; Rasheed, S.; Baloch, I.A. M-polynomials and topological indices for line graphs of chain silicate network and h-naphtalenic nanotubes. *J. Math.* **2021**, *2021*, 5551825. [CrossRef]
14. Baek, J.; Lee, D.B.; Hwang, S.J. Learning to extrapolate knowledge: Transductive few-shot out-of-graph link prediction. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 546–560.

15. Wang, C.; Zhang, M.; Ma, W.; Liu, Y.; Ma, S. Make it a chorus: Knowledge-and time-aware item modeling for sequential recommendation. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, Xi'an, China, 25–30 July 2020; pp. 109–118.
16. 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; pp. 353–362.
17. 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 2013 (NIPS '2013), Lake Tahoe, NV, USA, 5–10 December 2013; pp. 2787–2795.
18. 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.
19. Wang, X.; Wang, D.; Xu, C.; He, X.; Cao, Y.; Chua, T.S. Explainable reasoning over knowledge graphs for recommendation. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 5329–5336.
20. 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.
21. Wang, H.; Zhang, F.; Wang, J.; Zhao, M.; Li, W.; Xie, X.; Guo, M. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In Proceedings of the 27th ACM international Conference on Information and Knowledge Management, Turin, Italy, 22–26 October 2018; pp. 417–426.
22. 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, 2–7 October 2018; pp. 297–305.
23. 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, Xi'an, China, 25–30 July 2020; pp. 219–228.
24. Wang, H.; Zhao, M.; Xie, X.; Li, W.; Guo, M. Knowledge Graph Convolutional Networks for Recommender Systems. In Proceedings of the WWW '2019, San Francisco, CA, USA, 13–17 May 2019; pp. 3307–3313.
25. Rendle, S.; Freudenthaler, C.; Gantner, Z.; Schmidt-Thieme, L. BPR: Bayesian personalized ranking from implicit feedback. *arXiv* **2012**, arXiv:1205.2618.
26. 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.
27. 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; pp. 950–958.
28. Kingma, D.P.; Ba, J. Adam: A Method for stochastic optimization. In Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015), San Diego, CA, USA, 7–9 May 2015.