applied sciences

MDPI

Article

Neighborhood Aggregation Collaborative Filtering Based on Knowledge Graph

Dehai Zhang [1], Linan Liu [1], Qi Wei [1], Yun Yang [1,*], Po Yang [2] and Qing Liu [1]

[1] School of Software, Yunnan University, Chenggong, Kunming 650000, China; dhzhang@ynu.edu.cn (D.Z.); liulinan@mail.ynu.edu.cn (L.L.); weiqi199704@163.com (Q.W.); liuqing@ynu.edu.cn (Q.L.)

[2] Department of Computer Science, Sheffield University, Sheffield S1 1DA, UK; po.yang@sheffield.ac.uk

* Correspondence: yangyun@ynu.edu.cn

Received: 7 May 2020; Accepted: 27 May 2020; Published: 30 May 2020

Abstract: In recent years, the research of combining a knowledge graph with recommendation systems has caused widespread concern. By studying the interconnections in knowledge graphs, potential connections between users and items can be discovered, which provides abundant and complementary information for recommendation of items. However, most existing studies have not effectively established the relation between entities and users. Therefore, the recommendation results may be affected by some unrelated entities. In this paper, we propose a neighborhood aggregation collaborative filtering (NACF) based on knowledge graph. It uses the knowledge graph to spread and extract the user's potential interest, and iteratively injects them into the user features with attentional deviation. We conducted a large number of experiments on three public datasets; we verifyied that NACF is ahead of the most advanced models in top-k recommendation and click-through rate (CTR) prediction.

Keywords: recommendation system; collaborative filtering; knowledge graph; graph convolutional neural network; attention mechanism

1. Introduction

At present, many online recommendation services, such as e-commerce, advertising and social media, are based on historical interactions (purchases or clicks) to estimate the user's interest in the items. Collaborative filtering (CF) personalizes recommendations by analyzing users with similar behaviors [1]. For example, Jamali et al. [2] proposed a random walk model combining the trust-based and collaborative filtering approach for recommendation. Liu et al. [3] developed a Bayesian framework for predicting users' current news interests. Jamali et al. [4] proposed a social network recommendation method based on matrix decomposition. However, traditional collaborative filtering cannot solve the cold start problem effectively. Generally, researchers add some auxiliary information to solve such problems, such as social network [5], item's attributes [6], images [7] and heterogeneous network [8]. Among the various auxiliary information, the knowledge graph (KG) usually contains more abundant attributes and relationships about items.

The KG is a multi-relational directed graph composed of a large number of entities and relationships [9]. In recent years, researchers have proposed a number of public KGs (such as Freebase, DBpedia) and commercial KGs (such as the Google Knowledge Graph and Microsoft Satori). These KGs have been successfully applied to KG completion [10], data mining [11], question answering [12], text categorization [13] and many other fields.

At the same time, the combined research of KG and recommendation systems has received more and more attention [14]. As shown in Figure 1, the KG for movie scenario contains more comprehensive ancillary data, such as the background knowledge of the item and the relations between them. This approach can connect the interactions between items in multiple ways and display their potential relationships. More importantly, with rich links of items in KG, we can explore the interests of users in depth to discover their potential interests and complement the interaction of users and items.

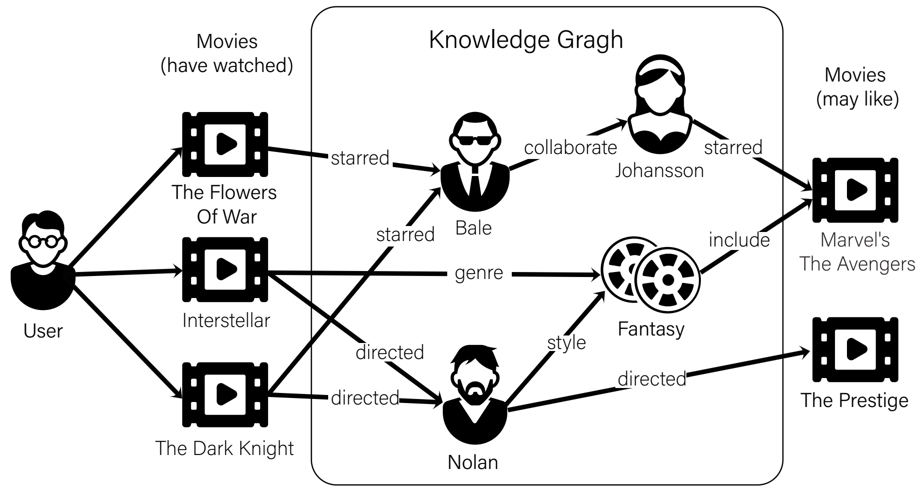

Figure 1. High-order link of entities in the movie knowledge graph.

It is the key that effectively extracts the auxiliary information in the KG into the recommendation. Recently, some works applied the graph convolutional neural network (GCN) to the recommendation system [15–18]. However, most of them are designed for user-item interaction graphs or user-item similarities graphs. There are some combined with KG, such as KGCN [18]. However, KGCN only extracts the information of items in the knowledge graph for recommendation, while ignoring the user–item interactions and personalized preferences.

In this work, we proposed neighborhood aggregation collaborative filtering (NACF) based on knowledge graph. The NACF applies GCN to KG to capture the potential relations between users and items and the users' personalized preferences. Specifically, we embed the user as a new entity type into KG. Their neighbors are items that the they have interacted with. The high-order neighborhood information of user is aggregated in a biased manner when evaluating the interest of a particular user-item pair. The advantages of this method are obvious. On the one hand, the users' potential interests can be discovered through the proliferation of neighboring entities. On the other hand, we use the attention mechanism to obtain the biased score of the neighboring entities under different user-item pairs, which satisfies individuation in the aggregation.

In summary, the main contributions of this paper are as follows:

- We propose a new neighborhood aggregation collaborative filtering based on a knowledge graph, which iteratively encodes the potential information of the knowledge graph into user features.
- We propose an attention mechanism that is adapted to neighborhood aggregation of the knowledge graph. It takes into account the personalized preferences and multiple associations in the information aggregation.
- We conduct a lot of experiments on three public datasets. The results show that NACF is significantly better than the existing methods in top-k recommendation and click-through rate (CTR) prediction. We also verified that NACF can keep firm performance in cold-start scenarios.

2. Related Work

2.1. Recommendation Based on Knowledge Graph

The existing recommendation schemes based on KGs can be divided into three categories.

(1) Embedding-based method [6,7,19]. It uses knowledge graph embedding (KGE) algorithm [9] to pre-process the recommended entity in the KG and input the embedding vector into a

recommendation framework. For example, DKN [19] combines entity embedding and title embedding as different initial features for news recommendation. CKE [7] combines knowledge graph embedding, text and item images into collaborative filter for unified recommendation. SHINE [6] designed a deep auto-encoder for embedding emotional network, social network and personal data network. However, knowledge graph embedding is usually used to extract the language association of each entity in the knowledge graph, and is more suitable for applications such as link prediction rather than the recommendation system.

(2) Path-based method [20–22]. In the knowledge graph of recommendation, various entities form complex connections, and multiple paths between them can be explored. These paths provide additional information for recommendation, especially on the interpretability. For example, PER [20] uses potential features of meta-path in the KG to represent the connectivity between users and items along different relationships. RKGE [21] and KPRN [22] introduce RNN to model multiple links of the user to items in the KG. The path-based approach makes it easy to take advantage of the connective features of KG. But these methods rely on manually set meta-paths and do not solve the cold-start problem very well. This is not conducive to practical application.

(3) Graph-based method [18,23,24]. Both the graph-based and path-based method are modeled by knowledge graph structure. The difference is that the graph-based method is not limited to the specific connection between entities, but regards KG as a heterogeneous network centered on a specific user or item. This method propagates from the center entity to extract the characteristics of the corresponding entity. RippleNet [23] proposed the idea of user interest diffusion, which spreads the users' interest characteristics by various entities of recorded interactions. KGCN [18] combines GCN with KG to extract embedded features of recommended item. Our method is also an example based on knowledge graph structure. But unlike the above algorithm, we embed the user into the KG and personalize the aggregation of user's characteristics.

2.2. Graph Convolutional Network

The graph convolutional network (GCN) is a deep learning method used to extract spatial features of topological graphs [25], which includes the spectral method [26] and non-spectral method [27]. The spectroscopy method defines the convolution calculation in the Fourier domain by calculating the eigen decomposition of the graph Laplacian. But the learning of the convolution kernel depends on the eigen decomposition of graph laplace matrix, which has certain requirements for the structure of graph and cannot be migrated to the model with other structure. The non-spectral method directly defines the convolutional calculation on the graph and takes into account neighbor nodes of different sizes while maintaining local convolution invariance.

Many of the existing recommendation methods import graph convolution network. PinSage [16] is a recommendation framework for advertising and shopping recommendations in social network. This algorithm combines random walks and graph convolution to generate node embedding that encompasses graph structure and node features. NGCF [17] uses the users' recorded interactions to generate graph and introduces the GCN to extract the potential features of user. KGCN [18] applies the GCN to the knowledge graph to generate the embedded features of the recommended entity.

In our work, we combine the user-item interactions with the corresponding KG, and introduce the idea of non-spectral GCN to dynamically extract the potential interests of the user.

2.3. Attention Mechanism

The brain receives a lot of external information all the time. When the brain receives the information, it will consciously or unconsciously use the attention mechanism to obtain more important information for itself. In recent years, the attention mechanism has been introduced into the fields of natural language processing, object detection, semantic segmentation, etc., and has achieved great effects.

The attention mechanism is also naturally applied to the recommendation system. DKN [19] builds a attention module to calculate the similarity between interactive items and recommended items. RippleNet [23] uses the similarity of the knowledge graph triples to calculate weights for different neighborhood entities. DIN [28] proposes a framework that uses the interest distribution to represent the diverse interest of user, and uses the attention mechanism to dynamically activate users' interests in different recommended items.

Our work is obviously different from the above. We not only consider the degree of user interest in the propagation of knowledge graph, but also consider the similarity between each entity in user's neighborhood and current recommended entity.

3. NACF Framework

3.1. Knowledge Graph Introduction

Our work is based on the knowledge graph for extraction of users' interest and items' recommendation. We define the KG for a particular recommendation scenario as *G = (E,R) = (h<e>, r<r>, t<e>)*, where *E* and *R* are entities and relations in the knowledge graph respectively and *h<e>*, *r<r>* and *t<e>* represent the head, relation and tail of a knowledge triple. In the recommendation scenario, the KG consists of a set of items and their related entities (such as item's attributes, external knowledge, etc.). For example, for the *«Titanic» (Titanic, film.film)*, the KG has a knowledge triple *(Titanic, film.film.star, Leonardo)* indicating that Leonardo DiCaprio is the star of the *«Titanic»*.

It is worth mentioning that our model is based on user-centered extraction through the knowledge graph. But the original knowledge graph does not include the user's entity. In this work, we embed the user as a new entity type into the knowledge graph and define a new relation type to connect the users to interacted item entity. For example, Mike has seen *«Dark Knight»*, so we added a new triple *(Mike,film.film.watch, The Dark Knight)* in the KG of movie. The following Table 1 shows the key symbols and their meanings in this paper.

Table 1. List of key symbols.

Symbol	Meaning
U={u_1,u_2...}	Set of users
I={i_1,i_2...}	Set of items
M	User-item interaction matrix
G={E,R}	Knowledge graph
E={e_i,e_j...}	Set of entities
R={$r_{(i,j)}$,$r_{(j,k)}$...}	Set of relations
$e_j^{[h]}$	Entity representation of e_j after the h-th aggregation update
N_{e_j}	The direct neighbor set of e_j in KG
$N_{e_j}^n$	the neighborhood of e_j in *n-hop*
$W_k^{(u,i)}$	The aggregate weight of each neighbor entity e_k under a given user-item pair

3.2. Model Framework Introduction

The NACF model has four parts of input: the user $u \in U$, the item $i \in I$, the user–item interaction matrix M and the corresponding knowledge graph G. Under the above conditions, NACF discovers the potential interest of users in KG and extracts hidden features between the various entities.

Specifically, given the user ID u, the item ID i and the neighbor set N_e of each entity in the KG, NACF predicts whether u has potential interest in the i that u has not contacted. The user's neighbors of KG is composed of item entities in user–item interaction record. The whole process is to learn the following prediction function:

$$y(u,i) = F(u,i|w,G_{<E,R>}) \tag{1}$$

where $y(u, i)$ represents the probability that user u will participate in item i, F represents the recommendation function and w represents the trainable parameter of F.

For each particular user–item pair, the neighbor features of the user are iteratively aggregated onto the entity e_u. It includes three steps: neighborhood construction, neighborhood aggregation and prediction.

(1) Neighborhood Construction

We take into account the entities in *n-hop* range of KG when extracting the users' features. As shown in Figure 2 left sub-picture, the neighbors in the *1-hop* range of e_u ($N^1_{e_u}$) are composed of directly connected entities (items with interactive record), the neighbors in the *2-hop* range ($N^2_{e_u}$) are composed of neighbors of directly connected entities in *1-hop* and so on. Assuming the *n-hop* neighbor set of e_u is $H^n_{e_u}$, neighborhood construction of e_u is as follows:

$$N^n_{e_u} = \sum_{k=1}^{n} H^k_{e_u} \tag{2}$$

Such a construction method has the obvious advantage that the model expands the physical neighborhood through the KG and can explore the potential interests of the user more broadly and deeply. The range of neighborhood construction has a great impact on the final results of feature extraction. The goal of this step is to find the right range of neighborhood to fully exploit the potential knowledge without introducing too much noise.

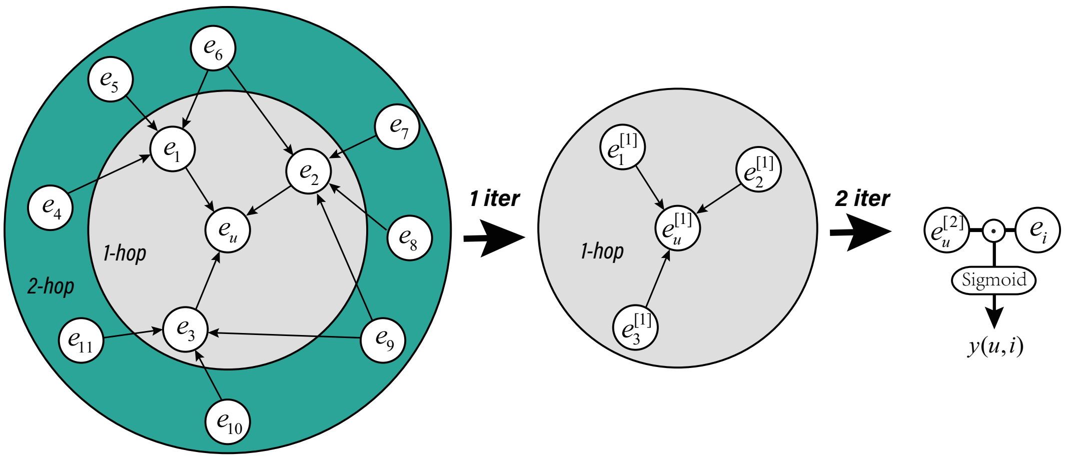

Figure 2. Neighborhood aggregation in KG.

(2) Neighborhood Aggregation

Before neighborhood aggregation, NACF initializes all entities and relations as trainable and random vectors (d-dimensions). If the neighborhood of entity includes *n-hop* (n layers), the aggregation of the entity is iterated n times in total. In the *h-th* iteration, all entities in the *n-h* layer perform aggregation of neighboring information and update the embedding vector. After one aggregation operation is completed, the updated representation of entity comes from the fusion of itself and the neighboring entities. We call such an aggregation operation as sub-aggregation Agg. Until the model is iterated n times (the knowledge graph neighborhood converges to e_u), the neighborhood aggregation of user is complete.

In a n layer neighborhood, e_u will update a total of *n-1* times and the final representation of e_u is $e_u^{[n]}$. For example, in a *2-hop* aggregation scenario in Figure 2, the first iteration updates the e_1 to $e_1^{[1]}$, and the e_u is updated to $e_u^{[1]}$, which we call the first-order representation of entity. In the second iteration, the $e_u^{[1]}$ is updated to $e_u^{[2]}$, which we call the second-order representation of entity. The specific sub-aggregation Agg will be described in the Section 3.3.

(3) Prediction

When the neighborhood aggregation of e_u is completed, the user's embedding entity $e_u^{[n]}$ and the item's entity e_i perform a point multiplication to generate a prediction score. Finally, this score will be normalized by the sigmoid function to the predicted click rate $y(u,i)$.

The NACF framework is shown in Algorithm 1:

Algorithm 1: NACF algorithm.

```
Input: $U$:set of user; $I$:Set of item; $M$:User-item interaction matrix; $G_{<E,R>}$:Knowledge graph;
    $n$:depth of Neighborhood; $d$:dimension of embedding;
Output: $F$:Prediction function
1  Embedding $U$ into $G_{<E,R>}$ and initialize $E, R, w$ as trainable parameters;
2  while NACF not converge do
3      for $(u,i) \in M$ do
4          $N_{e_u}^n = \sum_{k=1}^n H_{e_u}^k$
           for $h = 1, ..., n$ do
5              for $e_j \in N_{e_u}^{(n-h)}$ do
6                  $e_j^{[h]} \leftarrow Agg_{(u,i)}\left(N_{e_j}, e_j^{[h-1]}\right)$
7              end
8          end
9          $e_u \leftarrow e_u^{[n]}$
           Calculate predicted probability $y(u,i) = 1/\left(1 + e^{-e_u \cdot e_i}\right)$;
           Calculate loss value;
           Update parameters by gradient descent;
10     end
11     return $F$
12 end
```

3.3. Aggregation Process

The aggregation of user-item pairs (e_u,e_i) includes the sub-aggregation of all entities in the current neighborhood $N_{e_u}^n$. Sub-aggregation is a single process of aggregating information from directly connected neighbors to entity. As shown in Algorithm 1, given user id u and item id i, we define the *h-th* sub-aggregation process *Agg* of e_j as following:

$$e_j^{[h]} = Agg_{(u,i)}\left(N_{e_j}, e_j^{[h-1]}\right) \tag{3}$$

where $e_j \in N_{e_u}^{n-h}$ and $e_j^{[0]}$ is initial e_j.

As shown in Figure 3, we use the neighboring sub-aggregation process of the e_j as an example to illustrate the Agg in NACF. We use N_{e_j} to represent the collection of entities that are directly connected to e_j and assume that $N_{e_j} = (e_k, e_m, e_l)$. $r_{(j,k)}$ represents the relation between e_j and e_k. The corresponding link relations are $r_{(j,k)}, r_{(j,m)}, r_{(j,l)}$.

First we design a attention module to assign different weights to entities in the neighbor collection, since we believe that aggregation without distinction can introduce too much noise and is unreasonable. In NACF, we consider the degrees of users' interest in different relations and the similarity between neighboring entities and recommended entity. In this way, the weights of different entities in the aggregation have been confirmed.

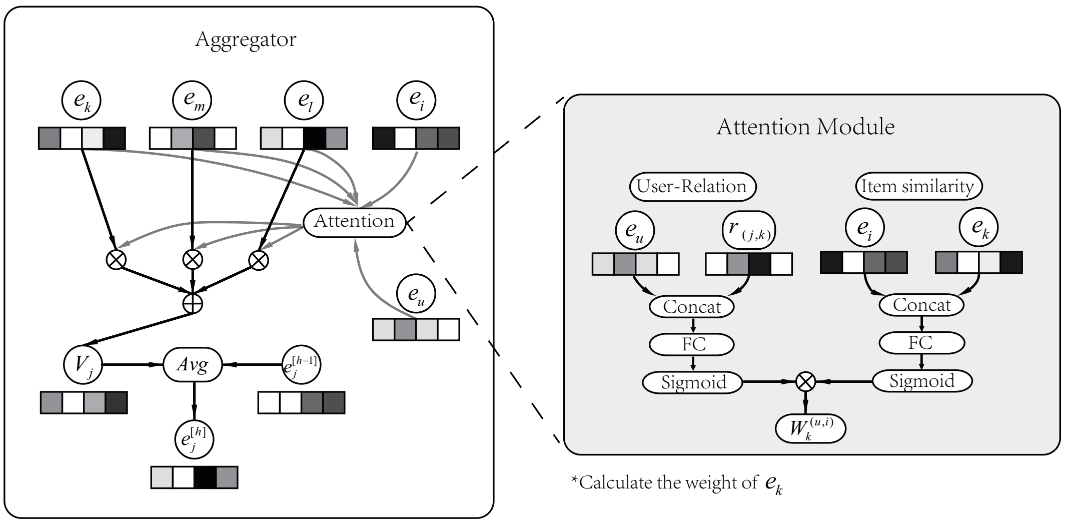

Figure 3. Neighborhood sub-aggregation process.

The attention module for calculating the weight is shown on the right sub-picture of Figure 3. For the weight calculation of the entity e_k, two parts of weight ($w_{(u,r)}$ and $w_{(e_i,e_k)}$) are calculated separately in the attention module. $w_{(u,r)}$ represents the user's attention to $r_{(j,k)}$. For example, one user may be more inclined to choose a particular singer when listening to music; another may care more about the style of the music. $w_{(e_i,e_k)}$ is the similarity between the current neighbor e_k and the recommended item e_i. We believe that entities with high similarity to recommended entity have a greater impact on user's choice. Finally, the weight of neighboring entities $W_{e_k}^{(u,i)}$ is two-part ($w_{(u,r)}$ and $w_{(e_i,e_k)}$) multiplication result. The process is as follows:

$$w_{(u,r)} = Softmax\left(w_1 \cdot \left(\text{concat}\left(e_u, r_{(j,k)}\right)\right) + b_1\right) \tag{4}$$

$$w_{(e_i,e_k)} = Softmax\left(w_2 \cdot \left(\text{concat}\left(e_i, e_k\right)\right) + b_2\right) \tag{5}$$

$$W_k^{(u,i)} = w_{(u,r)} \cdot w_{(e_i,e_k)} \tag{6}$$

where $w1$ and $w2$ are trainable vectors with $(2 \times d)$-dimensions. b_1 and b_2 are one-dimensional trainable vectors.

In order to integrate the neighborhood information of e_j, we perform an cumulative operation on the weighted neighbors to generate an aggregated vector V_j. The final step of the sub-aggregation is to calculate the mean of the original entity representation $e_j^{[h-1]}$ and neighborhood representation V_j, and update the entity representation of e_j to $e_j^{[h]}$. The update process of $e_j^{[h]}$ is as follows:

$$V_j = \sum_{e_k \in N_{e_j}} W_k^{(u,i)} \cdot e_k \tag{7}$$

$$e_j^{[h]} = \left(V_j + e_j^{[h-1]}\right)/2 \tag{8}$$

In the actual knowledge graph, there may be significant differences in the number of neighbors for each entity. To facilitate more efficient batch operation of the model, we extract a fixed-size neighbor set as a sample for each entity instead of its complete neighbor set. Specifically, the real neighborhood of the entity e_j in the knowledge graph is $N(e_j)$. We set its calculation neighbor set to $S(e_j)$. $S(e_j)$ samples K neighbors from $N(e_j)$, and K is a hyper-parameter.

3.4. Complete Loss Function

In order to better learn the NACF parameters and knowledge graph embedding representation, we designed the following complete loss function:

$$Loss = \sum_{u,i} \varphi \left(y(u,i), y(u,i)' \right) + \lambda_1 \|w\|_2^2 + \lambda_2 \left(\|E\|_2^2 + \|R\|_2^2 \right) \tag{9}$$

The loss function is divided into three parts. The first part represents the model prediction loss, and φ is the cross entropy loss function. The second part is the $L2$ regularization of the trainable parameter w in the model. The third part is the $L2$ regularization of the knowledge graph embedding, where E and R are the embedding vectors of all entities and relations in KG respectively. λ_1 and λ_2 in $Loss$ are configurable hyper-parameters. Because the above optimization problem is complicated, we use Adam [29] to iteratively optimize the loss function. We will discuss the choice of hyper-parameters in the experimental section.

4. Experiments

4.1. Datasets Introduction

We used three datasets of movie (MovieLens-20M), music (Last.FM) and restaurant (Dianping-Food) in the experiment. The datasets is as follows:

- MovieLens-20M is a widely used dataset for movie recommendation and contains approximately 20 million clear ratings (from 1 to 5) on the MovieLens website. The corresponding KG contains 102,569 entities, 499,474 edges and 32 relation types.
- Last.FM contains listening records for 2000 users in the online music system. The corresponding KG contains 9366 entities, 15,518 edges and 60 relation types.
- Dianping-Food comes from Dianping.com, which contains more than 10 million user and restaurant interaction records (including clicks, purchases, etc.), with approximately 2 million users and 1362 restaurants participating. The corresponding KG contains 28,115 entities, 160,519 edges and seven relation types.

The knowledge graph corresponding to the datasets is derived from the pre-processing knowledge graph disclosed by KGCN [18] and KGNN-LS [30]. These works use Microsoft Satori to build KG for movie and music datasets. The knowledge graph of Dianping-Food comes from the brain of Meituan Brain which is the internal knowledge graph built by Meituan for dining and entertainment. In data preprocessing, we removed the items that are not in the knowledge graph. Before NACF starts training, we initialize all entities and relations as trainable and random vectors. They are constantly adjusted during training until convergence.

Since the MovieLens-20M and Last.FM are rating feedback, we convert them to click-through rate feedback. For movie, there are 6783276 records with 4–5 points and 6,718,346 records with 1–3 points. So we choose four as the click-through rate to achieve a better samples distribution. The ratings of the music are too sparse (from 1 to 352,698) to be used as a criterion for evaluating user interest. Thus, we set all the music that the user clicked as positive samples. To avoid a large gap between the number of positive and negative samples, we used negative sampling during training to randomly select negative samples from items that were not interacted by users in the datasets until the numbers of positive and negative samples were the same.

The statistics of these three data sets are shown in Table 2.

Table 2. Statistics of the datasets.

	MovieLens-20M	Last.FM	Dianping-Food
#Users	138,159	1872	2,298,698
#Items	16,954	3846	1362
#Interaction	13,501,622	42,346	23,416,418
#Entities	102,569	9366	28,115
#Relations	32	60	7
#KG triples	499,474	15,518	160,519

4.2. Experimental Setup and Parameters

We selected two classical recommendation algorithms and three recommendation algorithms combined with KG as the baseline to compare the performance of NACF on the three datasets. The baselines are as follows:

- SVD [31] is a classic CF-based algorithm that uses inner product to model user-item interactions.
- LibFM [32] is a feature-based factorization model widely used for CTR prediction. In this experiment, TransR [33] was used to learn entity embedding for LibFM.
- CKE [7] combines collaborative filtering with knowledge embedding, text embedding, and item image embedding in a unified Bayesian framework. In this experiment, we represent each item only as an embedded vector learned by the TransR because the text descriptions and images in the datasets are not available.
- RippleNet [23] is a hybrid approach that uses KG information to assist recommendation, which spreads the user's potential preferences on KG for CTR recommendations. In this work, we use the TransE algorithm [34] to learn entity embedding.
- KGCN [18] is a recommendation algorithm that uses GCN to convolve item features, which makes use of the knowledge graph to extract item potential associations.

The code of NACF is implemented under Python 3.7, tensorflow-gpu 1.12.0 and NumPy 1.15.4. Experimental hardware environment includes AMD R5 2600, GTX 1070 and 16G memory. We set the ratio of training, evaluation and testing sets to 6:2:2 and manually adjusted the hyper-parameters of the NACF according to area under curve (AUC). Both the λ_1 and λ_2 in the loss function were set to 0.001. The hyper-parameters of NACF are shown in Table 3.

Table 3. Hyper-parameter settings.

	MovieLens-20M	Last.FM	Dianping-Food
#Neighbor	4	4	4
#Dimension	32	64	32
#hop	2	1	2
#Batch_size	1024	256	1024
#Epoch	10	10	10
#learning rate	2×10^{-2}	5×10^{-4}	2×10^{-2}

4.3. Experimental Results

In order to verify the performance of NACF, we compared NACF with baselines in top-k recommendation and CTR prediction. In the top-k recommendation, the trained model selects K items with the highest click rate for each user of the testing set, and evaluates the selected items of the model with Recall@K Precision@K and NDCG@K ($K \in 1, 2, 5, 10, 50, 100$). In CTR, we use area under curve (AUC), accuracy (ACC) and F1 score as metrics to evaluate the performances of all models. All experiments are conducted four times; the average values of the metrics are calculated.

The performances of all models in the top-k recommendation and CTR prediction are shown in Figure 4 and Table 4. We can observe:

(1) As shown in Figure 4, NACF achieve optimal performance in the top-k recommendation of three datasets. Specifically, NACF outperforms the best of baselines by 16.48%, 10.62% and 24.88% in Recall@20 on the movie, music and restaurant sets, respectively. It also achieved 15.30%, 24.43% and 17.51% performance lead in Precision@10. In NDCG@10, NACF took the lead by 8.11%, 18.37% and 9.60% respectively. It shows that NACF can make good use of knowledge graph to assist recommendation in TOP-K scenarios.

(2) In the CTR prediction, Table 4 shows that NACF achieved leading performance in AUC, ACC and F1. But the improvement is not very great, especially in movie. For music, the improvement of metrics is better than for movie. It shows that NACF can solve the sparsity problem to a certain extent, since Last.FM is much sparser than Movielens-20M and Dianping-Food.

(3) The performances of NACF, KGCN and RippleNet are basically ahead of CKE, SVD and libFM. It is worth mentioning that KGCN and Ripple also use the multi-hop neighborhood of KG, which illustrates that extracting multi-hop information in KG is necessary for recommendation. The CKE model performed very poorly in our experiments. It proves that CKE cannot make full use of KG with KGE algorithm like transR.

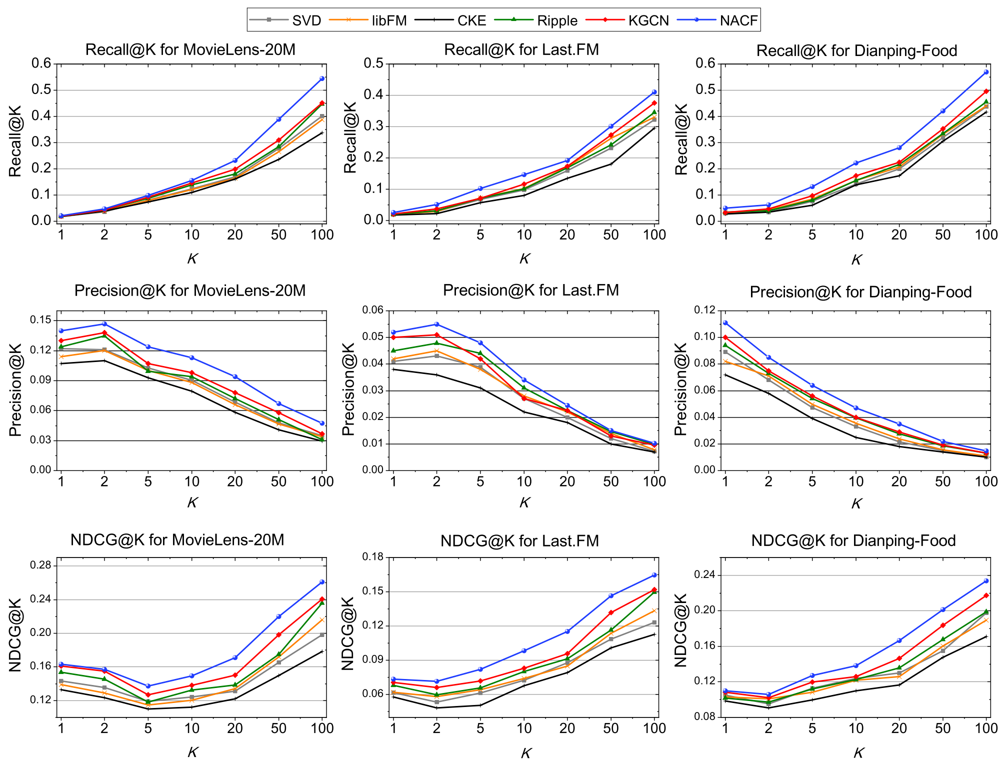

Figure 4. Performance in the top-k recommendation.

Table 4. Performance in CTR.

	Movielens-20M			**Last.FM**			**Dianping-Food**		
Method	**AUC**	**ACC**	**F1**	**AUC**	**ACC**	**F1**	**AUC**	**ACC**	**F1**
SVD	0.963	0.91	0.917	0.769	0.693	0.697	0.838	0.753	0.757
libFM	0.959	0.903	0.915	0.778	0.706	0.707	0.837	0.759	0.764
CKE	0.924	0.879	0.876	0.744	0.682	0.674	0.802	0.71	0.702
Ripple	0.968	0.921	0.914	0.78	0.702	0.709	0.833	0.751	0.758
KGCN	0.978	0.931	0.933	0.796	0.728	0.721	0.849	0.762	0.771
NACF	**0.983**	**0.940**	**0.938**	**0.821**	**0.746**	**0.741**	**0.882**	**0.792**	**0.795**
Improvement	0.51%	0.96%	0.53%	3.14%	2.47%	2.77%	3.87%	3.93%	3.11%

4.4. Performance Analysis

(1) Cold Start Problem

In this subsection, we verify that introducing KG into the recommendation can effectively alleviate the cold start problem. We gradually reduced the size of the MovieLens-20M training set from r = 100% to r = 20% (evaluation set and test set unchanged) to study the performances of models in cold start scenario. The results of AUC are shown in Figure 5. When r = 20%, the AUCs of the five baselines decreased by 7.5%, 5.9%, 2.8%, 4.8% and 1.9% respectively. The performance of NACF decreased by only 1.6%. This shows that NACF can achieve better result than baselines in a cold start scenario.

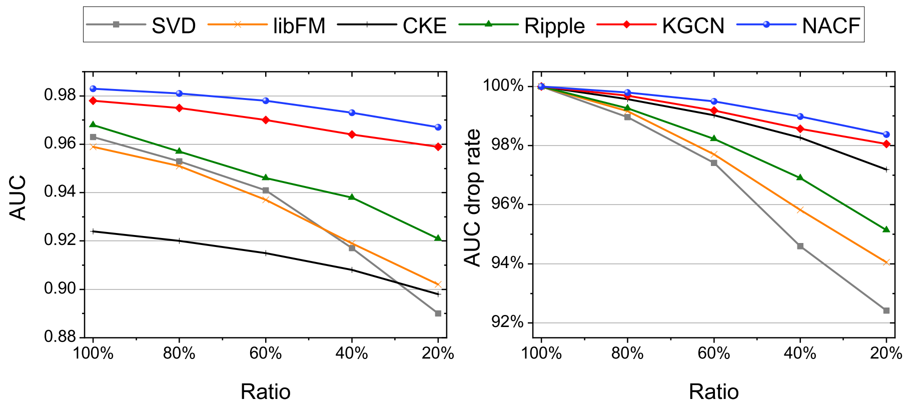

Figure 5. Model performances in cold start scenarios.

At the same time, we observe that the recommendation models based on KG performed better on the cold start problem than the classic models. This phenomenon verifies the effectiveness of KG information to alleviate the cold start problem.

(2) Effect of attention mechanism

Next, we study the effectiveness of the proposed attention module. We have mentioned that the attention module calculates the weights of the two parts weights $w_{(u,r)}$ and $w_{(e_i,e_k)}$ respectively in Section 3. $w_{(u,r)}$ represents the user's interest in the entities' current relation. $w_{(e_i,e_k)}$ is the similarity between the current neighbor e_k and the recommended item e_i. As shown in Figure 6, we observed the impact of different attention mechanisms on the Recall@*20* in three datasets.

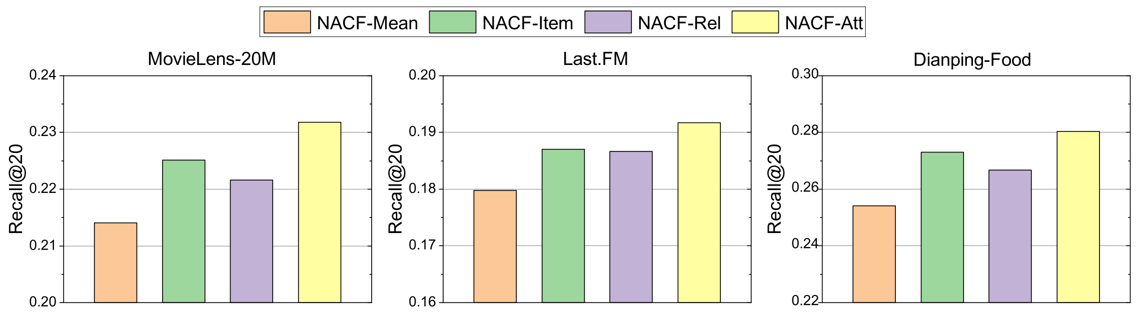

Figure 6. Performance under different attention mechanisms.

NACF-Mean means the NACF without attention module. In the sub-aggregation process, the calculation of V_j is to average the representation vector of neighbors. NACF-Rel and NACF-Item only use $w_{(u,r)}$ or $w_{(e_i,e_k)}$ respectively. NACF-Att represents the model that uses the full attention module. We observed that all models using attention mechanisms performed better than NACF-Mean. Moreover, the Recall@*20* performance of NACF-Item and NACF-Rel is lower than when them combined. We specifically observe the performance of the model in the MovieLens-20M.

Compared with NACF-Mean, NACF-Item and NACF-Rel increased by 7.47% and 4.95% respectively, and NACF-Att had a performance improvement of 10.35%. These results validate the effectiveness of the proposed attention module. In neighborhood information aggregation of KG, it is helpful that consider the user's interest in different relations and the similarity between entities.

(3) Impacts of different embedded dimensions

We analyzed the effects of different embedding dimensions of entity on NACF performance. The performances in the three datasets are shown in Table 5. When other parameters unchanged, the AUC of NACF will gradually rise with the increasing of the dimension d and achieve the optimal performance in *32*-dimension or *64*-dimension. But as the dimension gets larger, the performance gradually declines. That indicates that too large an embedding dimension may lead to over-fitting of NACF.

Table 5. Effects of different embedded dimensions on AUC.

Dim	**8**	**16**	**32**	**64**	**128**
Movielens-20M	0.98	0.981	**0.983**	0.981	0.978
Last.FM	0.818	0.821	0..827	**0.829**	0.825
Dianping-Food	0.878	0.88	**0.882**	0.879	0.876

(4) Impact of Different Neighbor Samples

We study the performance of NACF in different neighbor size by changing the number of neighboring samples K. As shown in the following Table 6, NACF gets the best performance in all three datasets when sampling four neighbors. It shows that too small a neighboring size K will make the model unable to extract enough knowledge information, while too large a K will introduce too much noise and cause a slight decline in performance.

Table 6. Impact of different neighbor samples on AUC.

Neighbors	**2**	**4**	**8**	**16**	**32**
Movielens-20M	0.98	**0.983**	0.982	0.981	0.978
Last.FM	0.825	**0.829**	0.827	0.823	0.819
Dianping-Food	0.878	**0.882**	0.88	0.879	0.876

(5) Impact of different neighborhood layers

We also analyze the effects of different sampling layers on the performance of the model. According to the NACF framework, more neighborhood layers will result in a geometric increase in the number of entities participating in calculation. The run time of NACF will also increase significantly. But the performance of the model does not increase gradually. As shown in Table 7, both the movie and restaurant achieve optimal performance when sampling layers is 2, and the Last.FM performs best when sampling layers is 1. Thus, the NACF needs a certain neighborhood to aggregate information. However, the larger sampling layers will introduce too many weakly related entities to participate in the aggregation and reduce the performance of the model.

Table 7. Effect of different sampling layers on AUC.

Layers	**1**	**2**	**3**	**4**
Movielens-20M	0.979	**0.983**	0.974	0.963
Last.FM	**0.829**	0.821	0.807	0.794
Dianping-Food	0.874	**0.882**	0.869	0.853

5. Conclusions

In this paper, we use neighborhood information of knowledge graph to aid the recommendation. We design a new recommendation framework NACF to incorporate the user's neighborhood information into collaborative filtering. The innovation of NACF is biased aggregation of neighborhood. It makes full use of structure of knowledge graph to extract features of user's interest, and introduces an attention mechanism to assign weights to different aggregate information. We conducted a large number of experiments on three public datasets and analyzed the rationality and effectiveness of using KG to assist the recommendation. It is verified that NACF has a performance level ahead of existing algorithms.

In future research, we will improve NACF and try to introduce other assistant information to understand users' behavior. For instance, semantic analysis combined with user comments, social networks, etc. We need try more ways of learning in feature mapping based on the knowledge graph, such as spectral GCN, graph attention network, more subtle attention mechanisms, etc. We will also focus on user online behavior analysis to achieve more effective and interpretable recommendations.

Author Contributions: Conceptualization, D.Z.; Data curation, Y.Y.; Investigation, P.Y.; Writing—original draft, L.L. and Q.W.; Writing—review and editing, L.L.; Resources, Q.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by (i) Natural Science Foundation China (NSFC) under Grant No. 61402397, 61263043, 61562093 and 61663046; (ii) Yunnan Provincial Young Academic and Technical Leaders Reserve Talents under Grant No. 2017HB005; (iii) Yunnan Provincial Innovation Team under Grant No. 2017HC012; (iv) Youth Talent Project of China Association for Science and Technology under Grant No. W8193209; (v) Science Foundation of Yunnan University under Grant No. 2017YDQN11; and (vi) Yunnan Provincial Science Research Project of the Department of Education under the Grant No.2018JS008.

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

References

1. Koren, Y.; Bell, R.; Volinsky, C. Matrix factorization techniques for recommender systems. *Computer* **2009**, *42*, 30–37. [CrossRef]
2. Jamali, M.; Ester, M. TrustWalker: A random walk model for combining trust-based and item-based recommendation. *Knowl. Discov. Data Min.* **2009**. [CrossRef]
3. Liu, J.; Dolan, P.; Pedersen, E.R. Personalized news recommendation based on click behavior. *Intell. User Interfaces* **2010**. [CrossRef]
4. Jamali, M.; Ester, M. A matrix factorization technique with trust propagation for recommendation in social networks. In Proceedings of the Fourth ACM Conference on Recommender Systems, New York, NY, USA, 26–30 September 2010.
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. Wang, H.; Zhang, F.; Hou, M.; Xie, X.; Guo, M.; Liu, Q. Shine: Signed heterogeneous information network embedding for sentiment link prediction. In Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, Los Angeles, CA, USA, 5–9 February 2018; pp. 592–600.
7. 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.
8. Vahedian, F.; Burke, R.; Mobasher, B. Multirelational Recommendation in Heterogeneous Networks. *ACM Trans. Web* **2017**, *11*, 1–34. [CrossRef]
9. Wang, Q.; Mao, Z.; Wang, B.; Guo, L. Knowledge graph embedding: A survey of approaches and applications. *IEEE Trans. Knowl. Data Eng.* **2017**, *29*, 2724–2743. [CrossRef]
10. Shi, B.; Weninger, T. Open-world knowledge graph completion. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018.

11. Wang, Y.; Jia, Y.T.; Liu, D.W.; Jin, X.; Cheng, X. Open web knowledge aided information search and data mining. *J. Comput. Res. Dev.* **2015**, *52*, 456–474.
12. Andreas, J.; Rohrbach, M.; Darrell, T.; Klein, D. Learning to compose neural networks for question answering. *arXiv* **2016**, arXiv:1601.01705.
13. Chen, J.; Hu, Y.; Liu, J.; Xiao, Y.; Jiang, H. Deep short text classification with knowledge powered attention. *arXiv* **2019**, arXiv:1902.08050.
14. Zhang, Y.; Chen, X. Explainable recommendation: A survey and new perspectives. *arXiv* **2018**, arXiv:1804.11192.
15. Wang, H.; Wang, N.; Yeung, D.Y. Collaborative deep learning for recommender systems. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, Australia, 10–13 August 2015; pp. 1235–1244.
16. 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, Macau, China, 14–17 April 2019; pp. 974–983.
17. Wang, X.; He, X.; Wang, M.; Feng, F.; Chua, T.S. Neural Graph Collaborative Filtering. *arXiv* **2019**, arXiv:1905.08108.
18. Wang, H.; Zhao, M.; Xie, X.; Li, W.; Guo, M. Knowledge graph convolutional networks for recommender systems. In Proceedings of the World Wide Web Conference, Lyon, France, 23–27 April 2018; pp. 3307–3313.
19. 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 International World Wide Web Conferences Steering Committee, San Francisco, CA, USA, 13–17 May 2019; pp. 1835–1844.
20. Yu, X.; Ren, X.; Sun, Y.; Gu, Q.; Sturt, B.; Khandelwal, U.; Norick, B.; Han, J. Personalized entity recommendation: A heterogeneous information network approach. In Proceedings of the 7th ACM International Conference on Web Search and Data Mining, New York, NY, USA, 24–28 February 2014; pp. 283–292.
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, 2–7 Obtober 2018, pp. 297–305.
22. 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.
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, Turin, Italy, 22–26 October 2018; pp. 417–426.
24. Tang, X.; Wang, T.; Yang, H.; Song, H. AKUPM: Attention-enhanced knowledge-aware user preference model for recommendation. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 1891–1899.
25. Zhou, J.; Cui, G.; Zhang, Z.; Yang, C.; Liu, Z.; Sun, M. Graph neural networks: A review of methods and applications. *arXiv* **2018**, arXiv:1812.08434.
26. Bruna, J.; Zaremba, W.; Szlam, A.; LeCun, Y. Spectral networks and locally connected networks on graphs. *arXiv* **2013**, arXiv:1312.6203.
27. Niepert, M.; Ahmed, M.; Kutzkov, K. Learning convolutional neural networks for graphs. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 19–26 June 2016; pp. 2014–2023.
28. Zhou, G.; Zhu, X.; Song, C.; Fan, Y.; Zhu, H.; Ma, X.; Yan, Y.; Jin, J.; Li, H.; Gai, K. Deep interest network for click-through rate prediction. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 1059–1068.
29. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
30. 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. *arXiv* **2019**, arXiv:1905.04413.
31. Koren, Y. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 426–434.
32. Rendle, S. Factorization machines with libfm. *ACM Trans. Intell. Syst. Technol.* **2012**, *3*, 57. [CrossRef]

33. 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.
34. Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; Yakhnenko, O. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2013; pp. 2787–2795.