information

Article

A Dynamic Convolutional Network-Based Model for Knowledge Graph Completion

Haoliang Peng * and Yue Wu

School of Computer Science and Engineering, Shanghai University, Shanghai 200444, China; ywu@shu.edu.cn
* Correspondence: penghaoliang@shu.edu.cn

Abstract: Knowledge graph embedding can learn low-rank vector representations for knowledge graph entities and relations, and has been a main research topic for knowledge graph completion. Several recent works suggest that convolutional neural network (CNN)-based models can capture interactions between head and relation embeddings, and hence perform well on knowledge graph completion. However, previous convolutional network models have ignored the different contributions of different interaction features to the experimental results. In this paper, we propose a novel embedding model named DyConvNE for knowledge base completion. Our model DyConvNE uses a dynamic convolution kernel because the dynamic convolutional kernel can assign weights of varying importance to interaction features. We also propose a new method of negative sampling, which mines hard negative samples as additional negative samples for training. We have performed experiments on the data sets WN18RR and FB15k-237, and the results show that our method is better than several other benchmark algorithms for knowledge graph completion. In addition, we used a new test method when predicting the Hits@1 values of WN18RR and FB15k-237, named specific-relationship testing. This method gives about a 2% relative improvement over models that do not use this method in terms of Hits@1.

Keywords: knowledge graph; knowledge graph completion; dynamic convolution network; knowledge graph embedding

Citation: Peng, H.; Wu, Y. A Dynamic Convolutional Network-Based Model for Knowledge Graph Completion. *Information* **2022**, *13*, 133. https://doi.org/10.3390/info13030133

Academic Editor: Vincenzo Moscato

Received: 28 December 2021
Accepted: 3 March 2022
Published: 4 March 2022

1. Introduction

Knowledge graphs are usually expressed in a highly structured form, where nodes denote the entities and edges represent different relations entities. This can be represented as a triples (h, r, t), where h and t stand for the head and tail entities respectively, and r represents the relation from h to t. At present, knowledge graphs have been widely used in many fields of artificial intelligence, such as automatic question answering [1], dialogue generation [2,3], personalized recommendation [4], and knowledge reasoning [5]. However, most open knowledge graphs, such as Freebase [6], Wikidata [7], and DBpedia [8], are constructed by automatic or semi-automatic methods. These graphs are usually sparse, and the implicit relationships among a large number of entities have not been fully mined. In Freebase, 71% of people do not have an exact date of birth, and 75% do not have nationality information. The incompleteness of knowledge graphs has become a major concern, and knowledge graph embedding [9] (KGE) is an effective solution to solve the problem of incompletion. Knowledge graph embedding generates low-dimensional embedding vectors of entities and relationships based on the existing triple relationships in the knowledge graph, and then inputs these low-dimensional embedding vectors into the score function, which can be a network to predict the missing relationships in the knowledge graph [10]. All models strive to make the positive triples score higher, and the negative triples score lower.

Recently, convolutional networks have been widely used to generate low-dimensional embedding vectors of entities and relationships in knowledge graphs, because convolutional networks can increase the interactions and capture interaction features between

head and relation embeddings. Empirical results have proved that increasing the number of interactions is beneficial to the knowledge graph embeddings task [11]. For example, ConvE [12] takes advantage of CNN and use 2D reshapings of head and relation embeddings to increase interactions. InteractE [11] takes advantage of circular convolution and feature permutation to increase interactions. However, most previous models have the following two problems. First, most previous models use the traditional convolutional kernel to capture interaction features between the head and relation embeddings and then they treat these interaction features uniformly. However, different interaction features have different contributions to the experimental results. Second, in the field of knowledge graph completion, traditional negative sampling is usually used to randomly replace the head entity or tail entity of the positive triples to obtain a certain number of negative triples before each round of model training, and then these positive triples and negative triples are put together to train the model. These random samples contain many easy negative samples; thus, this randomness reduces the model's ability to distinguish hard negative samples.

Based on the above observations, we propose a dynamic convolution network-based model using a method of mining hard negative samples for knowledge graph completion; we name it DyConvNE. In DyConvNE, for the first problem mentioned above, we use dynamic convolution kernel [13] instead of traditional convolution kernel. The dynamic convolutional network can assign different weights to these interaction features, allowing the network to pay more attention to more important interaction features and ignore unimportant interaction features. For the second problem mentioned above, we propose a hard negative triples sampling method. The model has no difficulty in distinguishing the easy negative samples, so these easy negative samples have limited training value for the model. Therefore, it is necessary to select a part of the negative samples that are difficult for the model to distinguish, called hard negative samples. Using these hard negative samples to train model, the difficulty during training can be increased, making the model pay more attention to details.

In particular, we propose a new test method when testing the Hits@1 values of WN18RR and FB15k-237. When we want to predict the tail entity of the triple $(h, r, ?)$, the traditional method is to put all the entities that appeared in the training set on the tail entity of the triple and then calculate their scores. However, there are many entities for which it is not necessary to calculate scores with triples. For example, relations are all parts of speech (verbs, adverbs, adjectives, etc.). If r is a verb, the tail entity of triple must be a verb word; it cannot be an adverb word or adjective word, etc. Motivated by the aforementioned observations, we only put the entities that appear after r in the training set on the tail entity of the triple and then calculate their scores. We name this method specific-relationship-testing.

Our contributions are as follows:

1. We propose a new model, called DyConvNE, based on a dynamic convolution network, which uses dynamic convolution to dynamically assign weights to the interaction features of the extracted entities and relationship embeddings;
2. We propose a method to mine hard negative samples and demonstrate the effectiveness of the method through ablation experiments;
3. We use specific-relationship-testing to obtain better performance on Hits@1;
4. We conduct some experiments to evaluate the performance of the proposed method. Experimental results demonstrate that our method obtains competitive performance on both WN18RR and FB15k-237.

2. Related Work

Recently, there have been many surveys of knowledge graph embedding, such as [14,15], and many different knowledge graph embedding models have been proposed. Models of knowledge graph embedding can be roughly divided into (1) decomposition-based models, (2) translation-based models, and (3) neural network-based models

RESCAL [16], NTN [17], and HOLE [18] models are typical decomposition-based models. Both RESCAL and NTN use tensor products. These tensor products capture rich interactions, but require a large number of parameters to establish a relational model, so calculations are very troublesome. To overcome these shortcomings, HOLE uses cyclic correlation of entity embedding to create a more effective and scalable representation decomposition.

In contrast, translation models such as TransE [19], DISTMULT [20], and ComplEx [21] have proposed simple models. TransE regards the relationship in the knowledge graph as a kind of translation vector between entities. For each triple (h, r, t), TransE uses the vector R of relation r as the translation between the head entity vector H and the tail entity vector T. We can also regard R as the translation of H from T. DISTMULT uses a bilinear diagonal model to learn embedding, which is a special case of using bilinear projection in NTN and TransE. DISTMULT uses weighted element dot products to model entity relationships. ComplEx uses complex embeddings and Hermitian dot products to learn embedding. These translation models are faster, require fewer parameters, and are relatively easy to train.

Recently, many neural network models have been proposed for knowledge graph embedding. Recently, CNN-based methods have been proposed to capture the interactive interaction features with parameter efficient operators, such as ConvE [12], ConvKB [22], and InteractE [11]. ConvE reshapes the initial input into a matrix form and then uses 2D convolution to predict links. It consists of a convolutional layer, a fully connected layer, and an inner product layer for final prediction. Using multiple filters to extract global relationships can generate different feature maps. The concatenation of these feature maps represents input triples. ConvKB is an improvement of ConvE. ConvKB does not need to reshape the input and captures more interactive features through convolutional layer. Compared with ConvE, which captures local relationships, ConvKB retains translation features and shows better experimental performance. InteractE proved that capturing more interactions between head and relation embeddings is beneficial to the final prediction result, so InteractE reorders the initial vector and reshapes the feature combination into many matrices and then puts those matrices into the convolutional layer to obtain more multi-feature interaction. However, InteractE only increases the interaction between head and relation embedding without taking into account the fact that the importance of different interaction features is not the same. Therefore, we propose to use dynamic convolution to assign weights of varying importance to interaction features to solve this problem.

3. Our Approach

In this section, we first describe the background and definitions used in the rest of the paper in Section 3.1 and introduce our model in Section 3.2. Then we introduce the dynamic convolutional network in Section 3.3, and the method of mining hard negative samples in Section 3.4. Finally, we introduce the loss function used in Section 3.5.

3.1. Definition

Knowledge Graph, Knowledge Graph Embedding and Negative Sampling are defined as follows.

Definition 1 (Knowledge Graph). *$\mathcal{G} = (\varepsilon, \mathcal{R})$, where $\varepsilon, \mathcal{R}$ indicate the entity set(nodes) and relationship set (edges) of the knowledge graph, respectively. A triple (h, r, t) is represented as the relation (edge) $r \in \mathcal{R}$ between head entity (node) $h \in \varepsilon$ and tail entity (node) $t \in \varepsilon$ in $\mathcal{G}$.*

Definition 2 (Knowledge Graph Embedding). *Knowledge graph embedding aims to learn an effective representation of entities, relations, and a scoring function f, which can be a network, such that for a given input triple, $v = (e_h, e_r, e_t)$, where e_h, e_r, e_t indicate h, r, t embedding vectors. $f(v)$ gives a v a higher score if v is valid. Therefore, we can predict the missing head entity h*

given query $(?, r, t)m$ *or tail entity* t *given query* $(h, r, ?)$*, with the learned entity and relation embedding and the scoring function.*

Definition 3 (Negative Sampling)**.** *Negative sampling generates negative triples by corrupting valid triples. Let* $K^+ = \{(h_j, r_j, t_j) \mid j = 1, 2, \cdots, N\}$ *denote the complete knowledge graph, where* (h_j, r_j, t_j) *represents the valid triple in the knowledge graph. Negative sampling produces sets of corrupted triples (negative triples)* $T' = \{(h, r, t') \mid t' \in \varepsilon, (h, r, t') \notin K^+\}$*. During model training, negative sampling takes a certain number of negative triples from* T' *for training.*

3.2. Our Model

The overall framework of our model is shown in Figure 1. Table 1 presents the parameters of our model. Our model contains three dynamic convolutional layers, a flatten layer, and a fully connected layer (FC). The first dynamic convolutional layer (DConv1) has 64 filters and the size of the kernel is 3 × 3. The next two dynamic convolutional layers (DConv2, DConv3) have 128 and 256 filters, and the two kernels are both 3 × 3. For all dynamic convolutional layers, apply the same padding and size of stride. The calculation rule of the output is defined in Section 3.3.

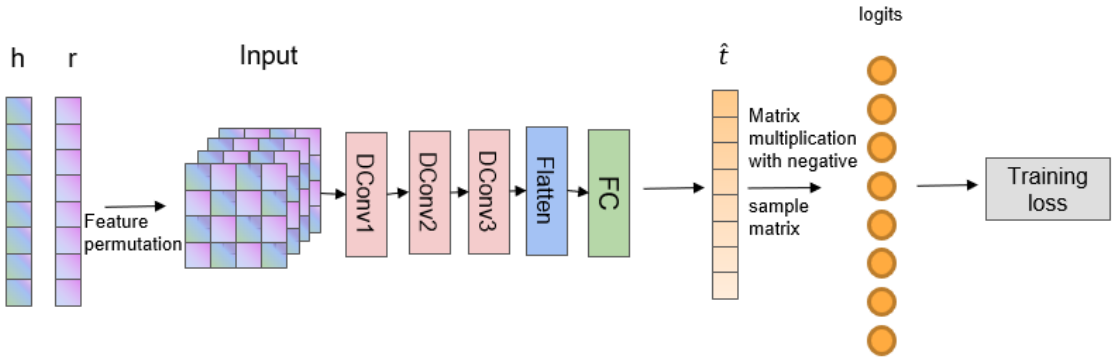

Figure 1. Overview of DyConvNE. In the DyConvNE model, the head and relation embeddings are first reshaped into four permutation matrices. Then, these matrices are used as an input to the dynamic convolutional layer. The resulting feature map tensors are flattened and projected in a 200-dimensional space and matched with all candidate object embeddings. Please refer to Section 3.2 for details.

Table 1. Parameter settings for our model.

Layer	Size of Stride	Number of Filters	Size of Kernel	Size of Padding	Output Size
Input	-	-	-	-	20 × 20 × 4
DConv1	1	64	3 × 3	1	20 × 20 × 64
DConv2	1	128	3 × 3	1	20 × 20 × 128
DConv3	1	256	3 × 3	1	20 × 20 × 256
Flatten	-	-	-	-	102,400
FC	-	-	-	-	200

For example, we have the valid triple $p = (h, r, t)$ and the input query $q = (h, r, ?)$. The embedding corresponding to h, r, t are e_h, e_r, e_t, respectively, and the dimensionality of embeddings is 200. First, we use the Chequer method proposed in InteractE to randomly scramble and combine the head embedding (e_h) and relationship embedding (e_r) into a matrix:

$$v_1 = \phi_{chk}(e_h, e_r) \tag{1}$$

where ϕ_{chk} denotes the Chequer method and $v_1 \in \mathcal{R}^{20\times20}$. We use the Chequer method four times to obtain four different matrices, denoted as v_1, v_2, v_3, and v_4. These matrices have the same size. The Chequer method has proven to be effective in increasing interaction between head embedding and relationship embedding [11]. Next, the four different

matrices are viewed as four channels of the input for the first dynamic convolutional layer (DConv1). We first use 3×3 dynamic convolution (proposed in Section 3.3) to capture interaction features and assign weights of varying importance to interaction features, which can be formulated as:

$$v_{3\times3}^{1} = r\Big([v_1||v_2||v_3||v_4] * \Omega^1\Big) \tag{2}$$

where $||$ denotes the concatenation operation, r denotes Relu activation function [23], $*$ denotes the convolutional operation, $[v_1||v_2||v_3||v_4] \in \mathcal{R}^{20\times20\times4}$, Ω^1 is the parameter of the first dynamic convolutional layer (DConv1), and $v_{3\times3}^{1} \in \mathcal{R}^{20\times20\times64}$ denotes the interaction features from the first dynamic convolutional layer (DConv1).

Second, we use two 3×3 dynamic convolutions to capture high-level interaction features and assign weights of varying importance to high-level interaction features:

$$v_{3\times3} = r\Big(r\Big(v_{3\times3}^{1} * \Omega^2\Big) * \Omega^3\Big) \tag{3}$$

where $v_{3\times3}^{1}$ is the input interaction features, Ω^2 is the parameter of the second dynamic convolutional layer (DConv2), Ω^3 is the parameter of the third dynamic convolutional layer (DConv3), r denotes Relu activation function [23], and $v_{3\times3} \in \mathcal{R}^{20\times20\times256}$ denotes the output interaction features from the third dynamic convolutional layer (DConv3).

Then, the final output interaction features $v_{3\times3}$ are flattened to 102,400 units and a fully connected layer is applied to obtain the predicted embedding of the given query:

$$e_{\hat{t}} = r\Big(vec(v_{3\times3})W^1\Big) \tag{4}$$

where $W^1 \in \mathcal{R}^{102{,}400\times200}$ is the parameter of the fully connected layer and r denotes Relu activation function [23].

Finally, in order to train the model, we need to sample some negative samples. First, we use our method (mining hard negative samples) to collect a small number of hard negative samples θ_1 of tail entity for the valid triple $(h,\ r,\ t)$. Second, we use the normal negative sampling method to collect a large number of negative samples θ_2 of tail entity. Therefore, we obtain a negative tail entity set $\theta = \theta_1 \cup \theta_2$. The label of t is 1 and the label of tail entity in θ_1 and θ_2 is 0. Then, we multiply the predicted tail embedding $e_{\hat{t}}$ with the valid tail embedding e_t and the tail embedding in the negative sample set θ to obtain the logits. We put logits into the loss function, intending to make the logits of the valid tail embedding e_t larger and larger, and the logits of the tail embedding obtained from negative sampling θ smaller and smaller. The number of negative samples and the setting of hyperparameters for our model are in Section 4.2.

3.3. Dynamic Convolution

Most previous convolutional network models like InteractE [11] and ConvE [12] use traditional convolutional filters to capture interaction features between the head and relation embeddings; then, they treat these interaction features uniformly. However, we think that the different interaction features should have different contributions to the experimental results. In this paper, we are inspired by the concept of dynamic convolution, which was proposed in [13] and was used in image processing. We propose to employ dynamic convolution to assign weights of varying importance to interaction features. Different from [13], which computes weights over convolutional kernels, we compute weights over convolutional filters (output channels).

The goal of dynamic convolution is to learn a group of filter weights, which can assign weights of varying importance to filters. We illustrate the overall framework of dynamic convolution in Figure 2, and Table 2 presents parameters of one dynamic convolutional layer, which has C_{out} filters and a kernel size of $s_1 \times s_2$. Dynamic convolution can build upon transformation mapping an input $X \in \mathcal{R}^{H\times W\times C_{\text{in}}}$ to feature maps (interaction features) $Y \in \mathcal{R}^{H'\times W'\times C_{\text{out}}}$. In the notation that follows, we use $V = [\text{con}\, v_1,\ \ldots,\ \text{con}\, v_{C_{\text{out}}}]$

to denote the learned set of convolutional filters, where $\mathrm{con}\, v_c \in \mathcal{R}^{s_1 \times s_2 \times C_{in}}$ refers to the parameters of the c-th filter and s_1, s_2 indicate the size of the convolutional kernel. Therefore, for the traditional convolutional layer, if the input is $X \in \mathcal{R}^{H \times W \times C_{\mathrm{in}}}$ we can obtain output $U = [u_1, u_2, \ldots, u_{C_{\mathrm{out}}}]$:

$$u_c = \mathrm{con}\, v_c * X = \sum_{s=1}^{C_{\mathrm{in}}} \mathrm{con}\, v_c^s * \mathbf{x}^s \tag{5}$$

where $*$ represents the convolution operation, $\mathrm{con}\, v_c = [\mathrm{con}\, v_c^1, \mathrm{con}\, v_c^2, \ldots, \mathrm{con}\, v_c^{C_{\mathrm{in}}}]$, $X = [\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^{C_{\mathrm{in}}}]$, and $u_c \in \mathcal{R}^{H' \times W'}$. $\mathrm{con}\, v_c^s \in \mathcal{R}^{s_1 \times s_2}$ is a 2D kernel representing a single channel of $\mathrm{con}\, v_c$ that acts on the corresponding channel of X. To simplify the notation, bias terms are omitted.

Table 2. Parameter settings for one dynamic convolutional layer.

Layer	Size of Stride	Number of Filters	Size of Kernel	Size of Padding	Output Size
Input	-	-	-	-	$H \times W \times C_{\mathrm{in}}$
Avg. Pool	-	-	-	-	$1 \times 1 \times C_{\mathrm{in}}$
Conv1	1	$C_{\mathrm{in}}/4$	1×1	0	$1 \times 1 \times C_{\mathrm{in}}/4$
Relu	-	-	-	-	$1 \times 1 \times C_{\mathrm{in}}/4$
Conv2	1	C_{out}	1×1	0	$1 \times 1 \times C_{\mathrm{out}}$
Softmax	-	-	-	-	$1 \times 1 \times C_{\mathrm{out}}$
Output	-	-	-	-	$H' \times W' \times C_{\mathrm{out}}$

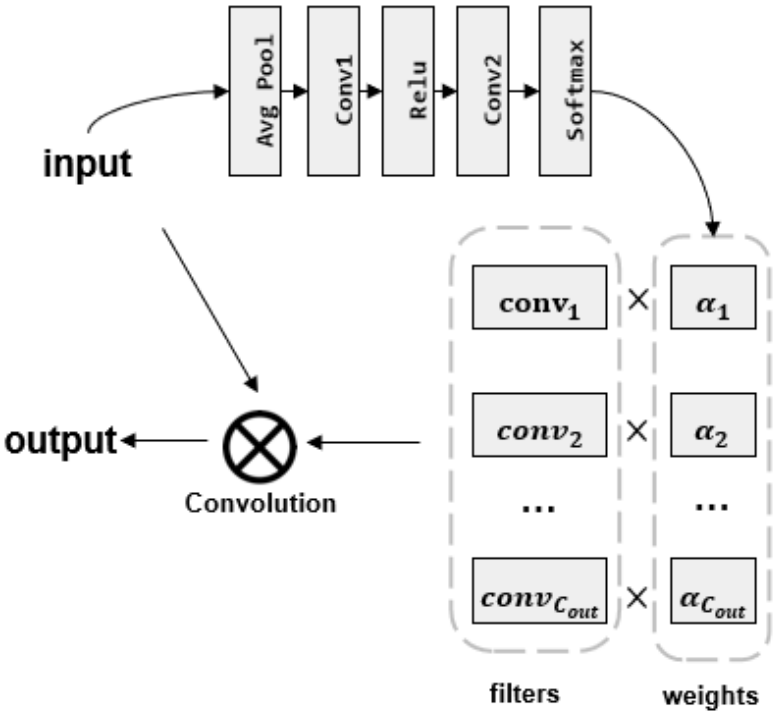

Figure 2. The overall framework of the dynamic convolutional layer.

For dynamic convolutional layer, we apply squeeze-and-excitation [24] to compute filter weights. Firstly, the input $X \in \mathcal{R}^{H \times W \times C_{\mathrm{in}}}$ is squeezed by the average pooling layer to become $X_1 \in \mathcal{R}^{1 \times 1 \times C_{\mathrm{in}}}$, such that the a-th element of $X_1 = [\mathbf{x}_1^1, \mathbf{x}_1^2, \ldots, \mathbf{x}_1^{C_{\mathrm{in}}}]$ is calculated by:

$$x_1^a = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} x^a(i,j) \tag{6}$$

where $x^a \in \mathcal{R}^{H \times W}$ is the a-th element in X. Then, we use two 1×1 convolutions (with a Relu between them) and softmax to generate normalized weights $\alpha = [\alpha_1, \alpha_2, \ldots, \alpha_{C_{out}}]$ for convolutional filters V:

$$\alpha = \sigma(V_2 * r(V_1 * X_1)) \tag{7}$$

where $\alpha \in \mathcal{R}^{1 \times 1 \times C_{out}}$, V_1 is the parameter of the Conv1, V_2 is the parameter of the Conv2, r denotes Relu activation function [23], and σ denotes the softmax function. Finally, the output $Y = [y_1, y_2, \ldots, y_k]$ of the dynamic convolutional layer can be formulated as:

$$y_c = (\text{conv}_c \times \alpha_c) * X \tag{8}$$

where $*$ denotes convolution, $y_c \in \mathcal{R}^{H' \times W'}$ is the c-th element in Y, and $\alpha_c \in \mathcal{R}^{1 \times 1}$ is the c-th element in α.

Therefore, the essence of dynamic convolution is to add different weights to the convolutional filters at different inputs, which can capture more important interaction features and allow the network to pay more attention to more important interaction features, instead of unimportant interaction features.

3.4. Mining Hard Negative Samples

Negative sampling generates negative triples by corrupting valid triples. Let $K = \{(h_i, r_i, t_i) \mid i = 1, 2, \cdots, N\}$ denote all valid triples in the knowledge graph, T^+ denote all entities in the knowledge graph, and $T_i = \{(h_i, r_i, t') \mid t' \in T^+, (h_i, r_i, t') \notin K^+\}$ denote all negative triples of the valid triple (h_i, r_i, t_i). We divide the negative triples T_i into two grades. If the model easily distinguishes between a negative triple $(h_i, r_i, t_a) \in T_i$ and the valid triple (h_i, r_i, t_i), we named such a negative triple an easy negative sample. Using a large amount of easy negative samples will not achieve the purpose of training a better model. If the model hardly distinguishes between a negative triple $(h_i, r_i, t_b) \in T_i$ and the valid triple (h_i, r_i, t_i), we named such a negative triple a hard negative sample. Hard negative samples can increase the difficulty of the model during training and allow the model to pay more attention to details.

Figure 3 shows the process involved in our method of mining hard negative samples. In order to effectively select hard negative samples, the method we use is to mine through the model itself. If model input is a valid triple (h_i, r_i, t_i), we firstly randomly select k negative triples from T_i and obtain $N_i = \{(h_i, r_i, n_c) \mid c = 1, \cdots, k, (h_i, r_i, n_c) \in T_i\}$. Next, we use valid triple (h_i, r_i, t_i) and N_i to train the initial model for one epoch and obtain a trained model M_1. Then, we use M_1 to predict the missing tail entity at $(\mathrm{h_i}, \mathrm{r_i}, ?)$, and obtain logits about the position of all entities T^+in this tail entity. We use logits to sort these tail entities from high to low: $(\mathrm{h_i}, \mathrm{r_i}, \mathrm{e_1})$, $(\mathrm{h_i}, \mathrm{r_i}, \mathrm{e_2})$, $\ldots$, $(\mathrm{h_i}, \mathrm{r_i}, \mathrm{e}_{|T^+|})$. We define that triples ranked higher than r_s are likely to be valid triples, triples ranked between r_s and r_t are hard negative triples (hard negative samples), and triples ranked lower than r_t are easy negative triples (easy negative samples). Therefore, we obtain the set of hard negative samples $P_i = \{(h_i, r_i, e_c) \mid c = r_s, r_s + 1, \cdots, r_t, (h_i, r_i, e_c) \in T_i\}$. We can use the hard negative samples obtained from this epoch model to train the next epoch model. Finally, we merge P_i to the negative sampling of (h_i, r_i, t_i) in the next epoch training. After that, this operation is repeated with the model obtained from the previous epoch of training.

In our method, k, r_s, r_t are hyperparameters whose values are different for WN18RR and FB15k-237. We select the k from {1000, 5000, 10,000}, the r_s-r_t of FB15k-237 from {10th-100th, 20th-100th, 30th-100th, 40th-100th}, and the r_s-r_t of WN18RR from {10th-200th, 20th-200th, 30th-200th, 40th-200th}. Finally, as shown in Table 3, the k is set to 1000 for FB15k-237 and 5000 for WN18RR and the r_s-r_t is set to 30th–100th for FB15k-237 and 20th–200th for WN18RR.

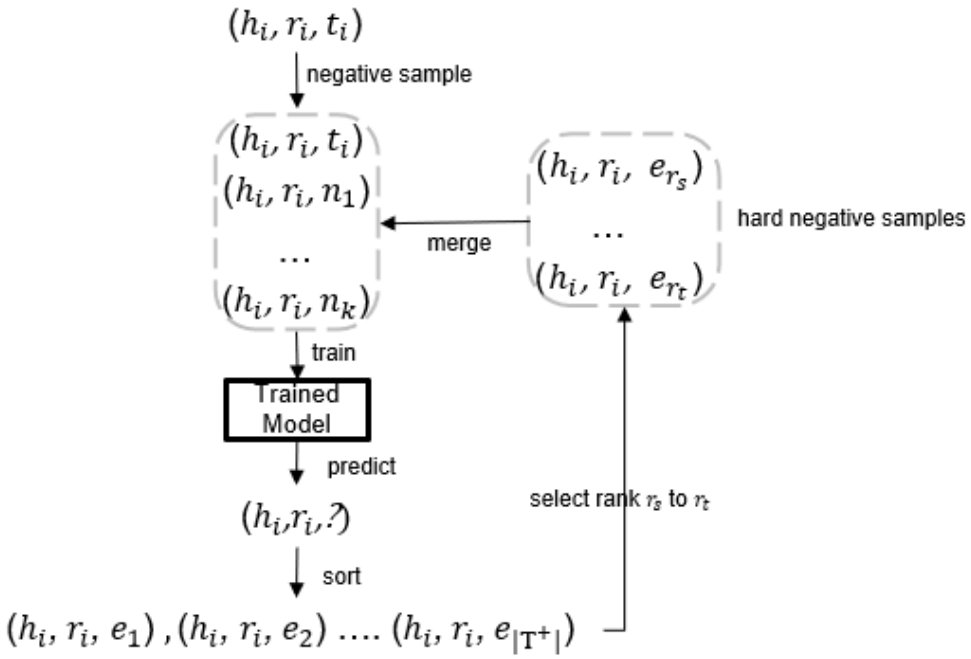

Figure 3. Process involved in mining hard negative samples.

Table 3. Hyperparameter settings for mining hard negative samples on FB15k-237 and WN18RR.

Parameter	Value	
	FB15k-237	WN18RR
k	1000	5000
r_s	30th	20th
r_t	100th	200th

3.5. Training Objective

For training the model parameters, we apply a logistic sigmoid to the logits of the scores of (h, r, t), and use the Adam optimizer [25] to train DyConvNE by minimizing the following binary cross-entropy loss:

$$\mathcal{L}(p, l) = -\frac{1}{N}\sum_i (l_i \cdot \log(p_i) + (1 - l_i) \cdot \log(1 - p_i)) \quad (9)$$

where p is the prediction and l is the label. For valid triples, we define its label as 1; for negative triples, it is defined as 0.

4. Experiments

In this part, we apply two public datasets, FB15k-237 and WN18RR, to validate the effectiveness of our proposed DyConvNE model for knowledge graph completion. Firstly, we give a detailed description of the datasets in Section 4.1 and an experimental setup of our model in Section 4.2. Then, we compare with other models to demonstrate the better performance of our model in Section 4.3 and we show the experimental results of our specific-relationship-testing method in Section 4.4. Finally, we present a case study in Section 4.5, an ablation study in Section 4.6, and a hard negative sampling study in Section 4.7.

4.1. Datasets

To evaluate our proposed approach, we used two benchmark datasets: WN18RR [12] and FB15K-237 [26]. WN18RR and FB15k-237 are the subsets of WN18 and FB15k after

removing the inverse relations, respectively. Previous works suggest that the task of knowledge graph completion in WN18 and FB15K suffers from the problem of inverse relations, whereby one can achieve state-of-the-art results using a simple reversal rule-based model [11]. Therefore, corresponding subset datasets WN18RR and FB15k-237 were created to resolve the reversible relation problem in WN18 and FB15K. Table 4 shows the summary statistics of the datasets. The WN18RR dataset has 40,943 entities, 11 relations, and 86,835 triples. The FB15k-237 dataset has 14,541 entities, 237 relations, and 272,115 triples.

Table 4. Statistics of the datasets.

Dataset	$\|\varepsilon\|$	$\|\mathcal{R}\|$	Triples		
			Train	Valid	Test
FB15k-237	14,541	237	272,115	17,535	20,466
WN18RR	40,943	11	86,835	3034	3134

4.2. Experimental Setup

In the current epoch of model training, we use the model that has been trained in the previous epoch to sample a certain number of hard negative samples for each valid triple (h, r, t). We obtain the hard negative sampling set of head entity θ_1 and the hard negative sampling set of tail entity θ_2. Then, we use the normal negative sampling method to collect a certain number of negative head entity sets θ_3 and negative tail entity sets θ_4. We replace the head entity and tail entity in the original triple (h, r, t) with the entities obtained by negative sampling to obtain the two replaced triple sets $(n_h,\ r,\ t)$, where $n_h \in \mathrm{h} \cup \theta_1 \cup \theta_3$, and $(h,\ r,\ n_t)$l where $n_t \in \mathrm{t} \cup \theta_2 \cup \theta_4$. The triple sets are put into the model for training, and label smoothing is used with a parameter of 0.1 for the training label. The parameters of our model are selected via grid search according to the MRR on the validation set. We select the initial learning rate from $\{0.001,\ 0.01,\ 0.1,\ 0.0002,\ 0.0001\}$, the dimensionality of embedding from $\{100,\ 200\}$, and the batch size from $\{64,\ 128,\ 256,\ 512\}$.

As shown in Table 5, for the FB15k-237 dataset, we use the Adam optimizer [25]. The initial learning rate we used was 0.001, and the dimensionality of embedding was 200. The learning rate decay strategy was used to decay 0.005 every 150 rounds. We trained the model up to 500 epochs with a batch size of 128 and a number of negative samples of 1070, including 70 hard negative samples. The selection range of hard negative samples was from the 30th to the 100th among the ranked entities. For the WN18RR dataset, the initial learning rate we used was 0.001, and the dimensionality of embedding was 200. The learning rate decay strategy was used to decay 0.005 every 150 rounds. We trained the model up to 500 epochs with a batch size of 256 and a number of negative samples of 5180, including 180 hard negative samples. The selection range of hard negative samples is from the 20th to the 200th among the ranked entities.

Table 5. Hyperparameter settings of our model.

Parameter	Value	
	FB15k-237	WN18RR
Learning rate	0.001	0.001
Epoch	500	500
Batch size	128	256
The dimensionality of embedding	200	200

4.3. Main Results

We follow the "Filtered" setting protocol [19] to evaluate our model, i.e., ranking all the entities excluding the set of other valid entities that appeared in training, validation, and test sets. We use Mean Reciprocal Rank (MRR), Mean Rank (MR), and Hits@1 and

Hits@10 indicators to evaluate our model. Note that lower MR, higher MRR, and higher Hits@1 or Hits@10 indicate better performance.

We have compared our results with various advanced methods: TransE [19], DisMult [20], ComplEx [21], ConvE [12], ConvKB [22], CACL [27], SACN [28], InteractE [11]. The experimental results are summarized in Table 6. As shown in Table 6, DyConvNE gains significant improvements on FB15k-237 and WN18RR. On FB15k-237 dataset, the MRR of our model is 0.358, Hits@1 is 26.5, Hits@10 is 54.2, and MR is 181. On FB15k-237, our model obtains the best result on Hits@1 and Hits@10, and the second-best result on MRR and MR. On the WN18RR dataset, the MRR of our model is 0.474, Hits@1 is 43.5, Hits@10 is 55.2, and MR is 4531. On WN18RR, our model obtains the best result on Hits@1, MRR, and Hits@10, and a comparable result on MR. We conduct ablation experiments in Section 4.5 to further demonstrate the effectiveness of our proposed dynamic convolution and of mining hard negative samples.

Table 6. Experimental results on WN18RR and FB15K-237 test sets. Results are taken from the corresponding papers. Hits@N values are in percentage. The best score is in bold and second best score is underlined.

Models	WN18RR				FB15k-237			
	MR	**MRR**	**Hits@1**	**Hits@10**	**MR**	**MRR**	**Hits@1**	**Hits@10**
TransE [19]	**2300**	0.243	4.27	53.2	323	0.279	19.8	44.1
DisMult [20]	7000	0.444	41.2	50.4	512	0.281	19.9	44.6
ComplEx [21]	7882	0.449	40.9	53	546	0.278	19.4	45
CACL [27]	3154	0.472	-	54.3	235	0.349	-	48.7
SACN [28]	-	0.470	43.0	54.0	-	0.350	26.0	54.0
ConvE [12]	4464	0.456	41.9	53.1	245	0.312	22.5	49.7
ConvKB [22]	2554	0.248	-	52.5	257	**0.396**	-	51.7
InteractE [11]	5202	0.463	43.0	52.8	**172**	0.354	26.3	53.5
DyConvNE (ours)	4531	**0.474**	**43.5**	**55.2**	181	0.358	**26.5**	**54.2**

4.4. Specific Relationship Testing

We experimented with our specific-relationship-testing method on FB15k-237 and WN18RR. The experimental results are summarized in Table 7. The traditional testing method is to replace either h or t with other entities in ε to create a set of invalid triples for each valid test triple (h, r, t) and then rank the valid test triple (h, r, t) and invalid triples in ascending order of their scores. If the valid test triple ranks first, the prediction is correct, and the total accuracy rate (Hits@1) increases. Our specific-relationship testing method considers the entity pairs connected by different relationships to be domain-specific. For example, if the relation r is place-lived, its tail entity must be a place name and not a person's name, age, etc. Therefore, in our specific-relationship-testing method, we replace either h_1 or t_1 with entities that should have appeared in the head entity or tail entity of the triple with relation r_1 in the training se, to create a set of invalid triples for each valid test triple (h_1, r_1, t_1). Then, we rank the valid test triple (h_1, r_1, t_1) and invalid triples in ascending order of their scores. Special attention is required when using the specific-relationship-testing method, since only Hits@1 can be compared to the traditional testing method because Hits@1 represents the accuracy rate. Other evaluation metrics (MR, MRR, and Hits@10) are not comparable because of the change in the number of invalid triples.

Table 7. The result of Hits@1 after using specific-relationship-testing on FB15K-237 and WN18RR. Hits@1 values are in percentage. The best score is in bold.

Models	Wn18RR	FB15k-237
	Hits@1	**Hits@1**
TransE [19]	4.27	19.8
DisMult [20]	41.2	19.9
ComplEx [21]	40.9	19.4
SACN [28]	43.0	26.0
ConvE [12]	41.9	22.5
InteractE [11]	43.0	26.3
DyConvNE	43.5	26.5
DyConvNe-SR	**46.3**	**28.4**

In the experiment, we used the same trained model and different testing methods to predict the Hits@1 of FB15k-237 and WN18RR. DyConvNE uses the traditional testing method, and DyConvNE-SR uses the specific-relationship-testing method. As shown in Table 7, DyConvNE-SR obtained the highest Hits@1 on WN18RR and also the highest Hits@1 on FB15k-237. DyConvNE-SR achieved better results than DyConvNE, and DyConvNE-SR gained significant improvements of 2.8% in Hits@1 on WN18RR and 1.9% in Hits@1 on FB15k-237.

4.5. Case Study

To further analyze how the specific-relationship-testing method contributes to knowledge graph completion, we give two examples in Table 8. For the query (Pixar Animation Studios, artist, ?), target (Randy Newman) ranks fourth when using traditional testing methods, and first when using our method. The traditional testing method ranks John A. Lasseter, Pete Docter, and Andrew Stanton in the first, second, and third place. Still, these are animators, directors, and screenwriters, respectively, not artists. When using the specific-relationship-testing method, all predictions for the query are a set of entities with the type "artist". For the second example (?, profession, theatrical producer), target (Emanuel "Manny" Azenberg) ranks second when using the traditional testing method and first when using our method. The traditional testing method ranks The Shubert Organization first, but it is not a person and has no attribute of profession. When using our method, all predictions for the query are a set of entities with the type "person". These examples clearly show how our specific-relationship-testing method benefits the knowledge graph completion.

Table 8. Two examples of top five predictions for the given queries. Two test triples were selected from the FB15k-237 test data set. The target in top predictions is underlined.

Query and Target	Top Predictions	
	Traditional Testing Method	**Specific-Relationship-Testing Method**
Query: (Pixar Animation Studios, artist, ?) **Target**: Randy Newman	John A. Lasseter Pete Docter Andrew Stanton Randy Newman Walt Disney Pictures	Randy Newman Mike Patton Ziggy Marley AC/DC Blondie
Query: (?, profession, theatrical producer) **Target**: Emanuel "Manny" Azenberg	The Shubert Organization Emanuel "Manny" Azenberg Marvin Neil Simon Tony Kushner Arthur Asher Miller	Emanuel "Manny" Azenberg Marvin Neil Simon Tony Kushner Arthur Asher Miller John Patrick Shanley

4.6. Ablation Study

In the ablation experiment, we ensured that the experiment was performed on FB15k-237 and WN18RR under the condition that the hyperparameters of the model DyConvNE, such as the learning rate, remained unchanged. First, we removed the method of mining hard negative samples in the DyConvNE, replaced the dynamic convolution in the model with traditional convolution, and obtained a general model DyConvNE-conv. As shown in Table 9, the MR of DyConvNE-conv on FB15k-237 is 186, the MRR is 0.353, and the @10 is 53.9. The MR of DyConvNE-conv on WN18RR is 5455, the MRR is 0.44, and the @10 is 51.6. Then, we only removed the method of mining hard negative samples in the model in DyConvNE and retained the dynamic convolution to get the model DyConvNE-dyconv. It can be observed that this model is better than DyConvNE-conv's MR, MRR, and @10 on both FB15k-237 and WN18RR, which proves the effectiveness of the dynamic convolution operation. Then, our original model DyConvNE-dyconv-neg, which includes dynamic convolution and mining hard negative samples, was compared to the DyConvNE-dyconv model The MR dropped by 4, the MRR increased by 0.002, and the Hits@10 increased by 0.2% on FB15k-237. DyConvNE-dyconv-neg also achieved better results on WN18RR. The ablation study proves that mining hard negative samples can make the model perform better.

Table 9. The result of ablation study on FB15K-237 and WN18RR test sets. Hits@10 values are in percentage. The best score is in bold.

Models	FB15k-237			WN18RR		
	MR	**MRR**	**Hits@10**	**MR**	**MRR**	**Hits@10**
DyConvNE-conv	186	0.353	53.9	5455	0.44	51.6
DyConvNE-dyconv	185	0.356	54.0	4801	0.452	52.5
DyConvNE-dyconv-neg	**181**	**0.358**	**54.2**	**4531**	**0.474**	**55.2**

4.7. Hard Negative Sampling Study

We present the results of our model on FB15k-237 in terms of Hits@10 in Figure 4 for rank (r_s-r_t) $\in$ {0th- 0th, 10th–100th, 20th–100th, 30th–100th, 40th–100th}. We also present the results of our model on WN18RR in terms of Hits@10 in Figure 5 for rank (r_s-r_t) $\in$ {0th- 0th, 10th–200th, 20th–200th, 30th–200th, 40th–200th}. The results show that the different rank (r_s-r_t) have different effects on the two datasets. The rank 0th-0th denotes that the model only uses the traditional negative sampling method, which is equivalent to the DyConvNE-dyconv model in Section 4.6. Figures 4 and 5 show that using a suitable rank of mining hard negative samples can achieve better performance than models using the traditional negative sampling method. Our model achieved the best results on the FB15k-237 when the rank (r_s-r_t) was set to 30th–100th, and our model achieved the best results on the WN18RR when the rank (r_s-r_t) was set to 20th–200th.

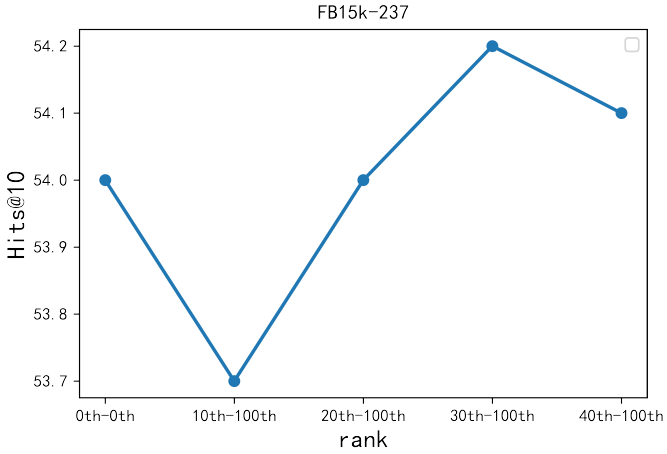

Figure 4. Experimental results on FB15k-237, evaluated in terms of Hits@10 for rank (r_s-r_t) $\in$ {0th-0th, 10th–100th, 20th–100th, 30th–100th, 40th–100th}.

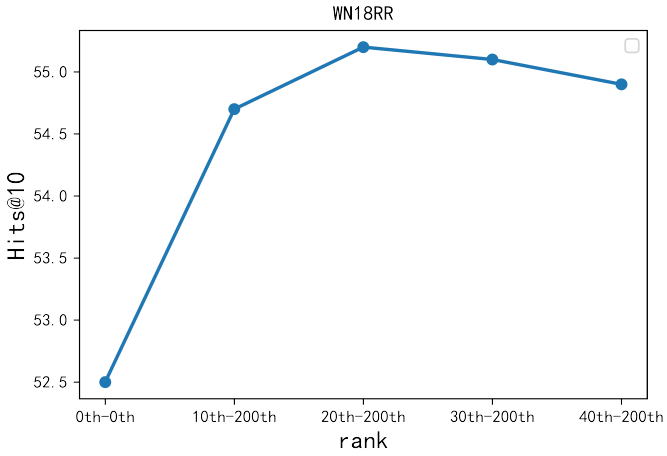

Figure 5. Experimental results on WN18RR, evaluated in terms of Hits@10 for rank (r_s-r_t) $\in$ {0th-0th, 10th-200th, 20th-200th, 30th-200th, 40th-200th}.

5. Conclusions

In this paper, we propose a new knowledge graph completion model named DyConvNE. Our model first employs dynamic convolution instead of traditional convolution to assign weights of varying importance to interaction features. Then, our model uses a new method of generating negative samples (mining hard negative samples) to increase the difficulty of model training. We proved the effectiveness of our model through experiments. The Hits@10 of our model reached 55.2% on the WN18RR dataset and 54.2% on the FB15k-237 dataset, which was better than the previous knowledge graph completion model. Finally, we propose a specific-relationship-testing method, which can reduce the number of candidate entities when testing. Our specific-relationship-testing method can improve Hits@1 by 2% on WN18RR and FB15k-237. In the future, we intend to extend our method to introduce attribute information of entities and relationships into the knowledge graph embedding to improve the accuracy of the knowledge graph embedding.

Author Contributions: H.P., theoretical study, analysis, findings, manuscript writing; Y.W., design review, manuscript review, and supervision. All authors discussed the results and contributed to the final manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: All the benchmarking datasets used in this study can be downloaded using the following URL: https://figshare.com/articles/dataset/KG_datasets/14213894 (accessed on 27 December 2021).

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

References

1. Huang, X.; Zhang, J.; Li, D.; Li, P. Knowledge graph embedding based question answering. In Proceedings of the 12th ACM International Conference on Web Search and Data Mining, Melbourne, Australia, 11–15 February 2019; pp. 105–113.
2. He, H.; Balakrishnan, A.; Eric, M.; Liang, P. Learning symmetric collaborative dialogue agents with dynamic knowledge graph embeddings. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, Vancouver, BC, Canada, 30 July–4 August 2017; pp. 1767–1776.
3. Madotto, A.; Wu, C.; Fung, P. Mem2seq: Effectively incorporating knowledge bases into end-to-end task-oriented dialog systems. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, Melbourne, Australia, 15–20 July 2018; pp. 1468–1478.
4. Zhang, F.; Yuan, N.; Nicholas, 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.
5. Chen, X.; Jia, S.; Xiang, Y. A review: Knowledge reasoning over knowledge graph. *Expert Syst. Appl.* **2020**, 141, 112948. [CrossRef]
6. Bollacker, K.; Evans, C.; Paritosh, P.; Sturge, T.; Taylor, J. Freebase: A collaboratively created graph database for structuring human knowledge. In Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, Vancouver, BC, Canada, 9–12 June 2008; pp. 1247–1250.
7. Vrandečić, D.; Krötzsch, M. Wikidata: A free collaborative knowledgebase. *Commun. ACM* **2014**, *57*, 78–85. [CrossRef]
8. Lehmann, J.; Isele, R.; Jakob, M.; Jentzsch, A.; Kontokostas, D.; Mendes, P.; Hellmann, S.; Morsey, M.; van Kleef, P.; Auer, S. Dbpedia—A large-scale, multilingual knowledge base extracted from wikipedia. *Semant. Web J.* **2015**, *6*, 167–195. [CrossRef]
9. Ji, S.; Pan, S.; Cambria, E.; Marttinen, P.; Philip, S. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *33*, 494–514. [CrossRef] [PubMed]
10. Dai, Y.; Wang, S.; Xiong, N. N.; Guo, W. A survey on knowledge graph embedding: Approaches, applications and benchmarks. *Electronics* **2020**, *9*, 750. [CrossRef]
11. Vashishth, S.; Sanyal, S.; Nitin, V.; Agrawal, N.; Talukdar, P.P. InteractE: Improving Convolution-Based Knowledge Graph Embeddings by Increasing Feature Interactions. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; AAAI Press: Palo Alto, CA, USA, 2020; pp. 3009–3016.
12. Dettmers, T.; Minervini, P.; Stenetorp, P.; Riedel, S. Convolutional 2d knowledge graph embeddings. In Proceedings of the AAAI Conference on Artificial Intelligence, Palo Alto, CA, USA, 4–9 February 2017; AAAI Press: Palo Alto, CA, USA, 2017; pp. 1811–1818.
13. Chen, Y.; Dai, X.; Liu, M.; Chen, D.; Yuan, L.; Liu, Z. Dynamic convolution: Attention over convolution kernels. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2020; pp. 11030–11039.
14. Rossi, A.; Barbosa, D.; Firmani, D.; Matinata, A.; Merialdo, P. Knowledge graph embedding for link prediction: A comparative analysis. *ACM Trans. Knowl. Discov. Data TKDD* **2021**, 15, 1–49. [CrossRef]
15. Wang, M.; Qiu, L.; Wang, X. A Survey on Knowledge Graph Embeddings for Link Prediction. *Symmetry* **2021**, *13*, 485. [CrossRef]
16. Nickel, M.; Tresp, V.; Kriegel, H.P. A Three-Way Model for Collective Learning on Multi-Relational Data. In Proceedings of the ICML, Washington, DC, USA, 28 June–2 July 2011; pp. 809–816.
17. Socher, R.; Chen, D.; Manning, C.; Ng, A. *Reasoning with Neural Tensor Networks for Knowledge Base Completion*; MIT Press: Cambridge, MA, USA, 2013; pp. 926–934.
18. Nickel, M.; Rosasco, L.; Poggio, T.A. Holographic Embeddings of Knowledge Graphs. In Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; AAAI: Phoenix, AZ, USA, 2016; pp. 1955–1961.
19. Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; Yakhnenko, O. Translating embeddings for modeling multi-relational data. *Adv. Neural Inf. Process.* **2013**, *26*, 2787–2795.
20. Yang, B.; Yih, W.; He, X.; Gao, J.; Deng, L. Embedding entities and relations for learning and inference in knowledge bases. In Proceedings of the ICLR (Poster), San Diego, CA, USA, 7–9 May 2015.
21. Trouillon, T.; Welbl, J.; Riedel, S.; Gaussier, É.; Bouchard, G. Complex embeddings for simple link prediction. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 20–22 June 2016; ICML: New York City, NY, USA, 2016; pp. 2071–2080.
22. Nguyen, D.Q.; Nguyen, T.D.; Nguyen, D.Q.; Phung, D.Q. A novel embedding model for knowledge base completion based on convolutional neural network. In Proceedings of the NAACL-HLT, New Orleans, LA, USA, 1–6 June 2018; pp. 327–333.
23. Xavier, G.; Antoine, B.; Yoshua, B. Deep sparse rectifier neural networks. In Proceedings of the AISTATS, Lauderdale, FL, USA, 11–13 April 2011; pp. 315–323.

24. Jie, H.; Li, S.; Gang, S. Squeeze-and-excitation networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018; pp. 7132–7141.
25. Kingma, D.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
26. Toutanova, K.; Chen, D.; Pantel, P.; Poon, H.; Choudhury, P.; Gamon, M. Representing Text for Joint Embedding of Text and Knowledge Bases. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, 17–21 September 2015; pp. 1499–1509.
27. Oh, B.; Seo, S.; Lee, L. Knowledge graph completion by context-aware convolutional learning with multi-hop neighborhoods. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management, Torino, Italy, 22–26 October 2018; pp. 257–266.
28. Shang, C.; Tang, Y.; Huang, J.; Bi, J.; He, X.; Zhou, B. End-to-End Structure-Aware Convolutional Networks for Knowledge Base Completion. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; AAAI Press: Palo Alto, CA, USA, 2019; pp. 3060–3067.