Applied Intelligence
https://doi.org/10.1007/s10489-021-02438-8

Novel translation knowledge graph completion model based on 2D convolution

Jianzhou Feng[1,2] · Qikai Wei[1,2] · Jinman Cui[1,2] · Jing Chen[1,2]

Accepted: 15 April 2021

Abstract
The knowledge graph completion task involves predicting missing entities and relations in a knowledge graph. Many models have achieved good results, but they have become increasingly complex. In this study, we propose a simple translation-based model that relies on the fact that the multiplication of subjects and relations is approximately equal to the object. First, we utilize embeddings to represent entities and relations. Second, we perform vector multiplication on subject embedding and relation embedding to generate a 2D matrix and achieve full fusion of embedding at the element level. Third, we adopt a convolutional neural network on the 2D matrix. Thereafter, we can generate feature maps, which are then spliced into a 1D feature vector. The feature vector is transformed into predicted object embedding through a fully connected operation. Finally, we use the scoring function to score the candidate triples. Experimental results strongly demonstrate that the translation knowledge graph completion model based on 2D convolution achieves state-of-the-art results compared with the baseline.

Keywords Knowledge graph completion · Convolutional neural network · Translation model · Entity relation matrix

1 Introduction

Knowledge graph G is a collection that features valid triples in the form of subject (s), relation (r), object (o), such that s, $o \in E$ and $r \in R$, where E is a set of entities, and R is a set of relations. Knowledge graphs are widely used as an effective resource in many applications, such as semantic searching and ranking [23], question answering [13], and machine reading comprehension [17]. Large-scale knowledge graphs, such as DBpedia, YAGO, and Freebase, still suffer from serious data sparseness and missing data problems. In Freebase, for example, 75% of character entities do not have nationality information, while 71% do not have an exact date of birth [6]. Knowledge graph completion has become a new research topic in recent years, producing the problem of incomplete information in knowledge graph.

The knowledge graph completion task aims to predict the missing entity or relation in a triple, i.e., inferring s given (r, o), inferring r given (s, o), or inferring o given (s, r). Figure 1 shows an example of a simple incomplete knowledge graph wherein the solid lines represent existing facts and the dotted lines represent information that must be inferred. For example, we need to predict the missing relation in (Huawei, ?, Apple Inc.) or the missing object in (Ren Zhengfei, IsHusbandOf, ?).

Many embedding models have been proposed for dealing with the knowledge graph completion task by learning the vector or matrix representations of entities and relations. TransE [3] is the most classic translation-based model that utilizes vectors to model entities and relations. It maps entities and relations to a fixed space. If (s, r, o) is true, then the distance between $s + r$ and o should be the closest;

✉ Qikai Wei
mrweiqk@163.com

Jianzhou Feng
fjzwxh@ysu.edu.cn

Jinman Cui
cuijinman@163.com

Jing Chen
xychenjing@ysu.edu.cn

1 School of Information Science and Engineering, Yanshan University, 066004 Qinhuangdao, China

2 Software Engineering Key Laboratory of Hebei Province, Yanshan University, 066004 Qinhuangdao, China

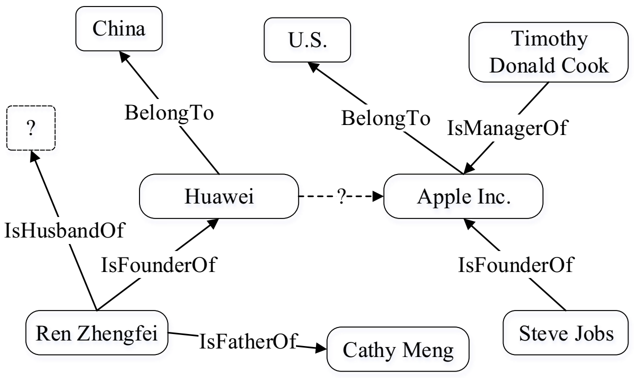

Fig. 1 Example of an incomplete knowledge graph

otherwise, the distance between $s + r$ and o is farther, and the mapping relationship of $s + r \approx o$ is constructed.

With the widespread application of deep learning in the knowledge graph field, convolutional neural networks (CNNs) have elicited sufficient attention. ConvE [5] solves the knowledge graph completion task by using a CNN. It divides the entire entity embedding and relation embedding into multiple fragments and then concatenates these fragments into a 2D matrix. Each element in the matrix represents either entity information or relationship information. Then, features are extracted from the matrix by using a 2D convolution operation. The features extracted in this manner have no practical meaning. Although ConvE has achieved impressive results, its reshaping and concatenation of embeddings are poorly interpretable. In addition, the features extracted via 2D convolution can be used after the entities and relations are fully fused.

Therefore, we propose the translation knowledge graph completion model based on 2D convolution (CTKGC), a new embedding model that constructs a mapping relationship of $s * r \approx o$ to achieve the knowledge graph completion task. It performs vector multiplication operations on 1D subject embedding and relation embedding to generate a 2D matrix that facilitates 2D convolution. Each element in the matrix is the product of the corresponding element in subject embedding and relation embedding, and thus, element-level fusion is achieved. The model uses the features extracted from the matrix to predict the object (o) better. Thus, we improve the interpretability of the algorithm process.

Our contributions in this study are as follows:

We present CTKGC, a new convolution-based translation model that constructs a mapping relationship of $s * r \approx o$ for knowledge graph completion, in contrast with TransE. We construct a matrix by fusing subject embedding and relation embedding at the element level to facilitate 2D convolution, avoiding the problem of 1D vectors being unable to achieve 2D convolution.

Compared with other models based on convolution methods, CTKGC has fewer parameters and the algorithm process is more interpretable.

We evaluate CTKGC on four benchmark datasets: FB15k-237 [20], WN18RR [5], KINSHIP [11], and Unified Medical Language System (UMLS) [4]. The experimental results show that CTKGC exhibits better performance than previous state-of-the-art (SOTA) models.

2 Related work

Many knowledge graph completion models have been proposed recently. These models can be divided into the following categories: translation-based, tensor factorization-based, and neural network-based knowledge graph completion methods.

A translation-based knowledge graph completion method regards the relation in fact triples as a translation between the subject and the object. TransE [3] is the most classic translation-based model. Other translation-based models, such as TransR [12] and TransD [8], extend TransE and use projection vectors or matrices to transform subject embedding and object embedding into the relation embedding space.

A knowledge graph completion model based on tensor factorization uses tensors to represent subject and relation, thereby factoring tensors. RESCAL [15] is an early model based on tensor factorization. It deals with knowledge graph completion by optimizing the scoring function that contains a bilinear product between subject and object vectors and a full rank relation matrix. Although RESCAL is a powerful model, it easily overfits during training because it has too many parameters. DistMult [22] is a special case of RESCAL with a diagonal matrix per relation, which reduces the possibility of overfitting. Its linear transformation performed on entity embedding is limited to a stretch. The binary tensor learned by DistMult is symmetric, and thus, DistMult cannot model asymmetric relations. ComplEx [21] extends DistMult to the complex domain. The subject embedding and object embedding of the same triple exhibit a complex conjugate relation that introduces asymmetry into tensor factorization, and thus, ComplEx can model asymmetric relationships and compensate for DistMult's deficiency, i.e., it can only model symmetric relationships. SimplE [9] applies the principle of canonical polyadic decomposition to perform vector training on the subject and the object to improve the model's interpretability.

A knowledge graph completion model based on a neural network can automatically extract features and use them to predict entities. HypER [1] uses a hypernetwork to generate a set of 1D relation-specific filters to process

Table 1 Scoring function $\psi_r(e^s, e^o)$ of knowledge graph completion, dimensionality of their relation parameters and space complexity

Model	Scoring function $\psi_r(e^s, e^o)$	Relation parameters	Space complexity
TransE [3]	$\Vert e^s + e^r - e^o \Vert_p$	$e^r \in \mathbb{R}^{d_r}$	$O(n_e d_e + n_r d_e)$
RESCAL [15]	$(e^s)^T e^r e^o$	$e^r \in \mathbb{R}^{{d_e}^2}$	$O(n_e d_e + n_r d_r^2)$
DistMult [22]	$\langle e^s, e^r, e^o \rangle$	$e^r \in \mathbb{R}^{d_e}$	$O(n_e d_e + n_r d_e)$
ComplEx [21]	$\mathrm{Re}(\langle e^s, e^r, \bar{e}^o \rangle)$	$e^r \in \mathbb{C}^{d_e}$	$O(n_e d_e + n_r d_e)$
ConvE [5]	$f(vec(f([\widetilde{e^s}; \widetilde{e^r}] * w))\mathrm{W})e^o$	$e^r \in \mathbb{R}^{d_r}$	$O(n_e d_e + n_r d_r)$
SimplE [9]	$\frac{1}{2}(\langle h_{e^s}, e^r, t_{e^o} \rangle + \langle h_{e^s}, e^{r^{-1}}, t_{e^o} \rangle)$	$e^r \in \mathbb{R}^{d_e}$	$O(n_e d_e + n_r d_e)$
CTKGC(proposed)	$f(vec([(e^s)^T \otimes e^r] * w))\mathrm{W})e^o$	$e^r \in \mathbb{R}^{d_r}$	$O(n_e d_e + n_r d_r)$

subject embedding, simplifying the interaction between subject embedding and relation embedding and reducing the number of model parameters. The neighbor node information of each subject plays an important role in the knowledge graph completion task. R-GCN [16] uses graph convolutional network to model relation data, regarding the adjacency matrix of each entity as an influence factor in the prediction process. R-GCN is used as an encoder and DistMult or ConvE is used as a decoder, considerably improving the accuracy of knowledge graph completion. In consideration of the fact that the subject embedding and relation embedding used by the knowledge graph completion model in predicting an object are fixed, A2N [2] can automatically combine subjects and their correlation graph neighbor nodes, changing the subject embedding with a different query $(s, r, ?)$ to generate a new subject embedding and predict an object more accurately. Moreover, CrossE [23] models the crossover interactions of entities and relations by learning an interaction matrix and generating more reliable explanations.

The aforementioned methods have achieved good results; however, a certain gap must still be filled before they can be applied in real life. Improving the performance of knowledge graph completion remains an important task. The use of deep learning to solve the knowledge graph completion task is a good strategy because features can be extracted automatically. We extract features from a matrix that combines complete subject information and relation information, such that the result is closer to the true object.

Neural knowledge graph completion models can be regarded as multilayered neural networks composed of an encoding component and a scoring component. Given an input triple (s, r, o), the encoding component uses embedding e^s and e^r to denote the embedding of subject s and relation r, respectively, and e^o to denote the embedding of object o. In the scoring module, $\psi_r(e^s, e^o)$ is a scoring function whose result represents the score of the corresponding triple (s, r, o).

We summarize several scoring functions for knowledge graph completion that have achieved SOTA results, as indicated in Table 1. d_e and d_r are the dimensionalities of entity embedding and relation embedding; and n_e and n_r denote the number of entities and relations, respectively. $\bar{e}^o \in \mathbb{C}^{d_e}$ denotes the complex conjugate form of e^o; $*$ is a convolution operation; $\widetilde{e^s}$ and $\widetilde{e^r}$ denote the 2D deformation of e^s and e^r, respectively; f denotes a nonlinear function; h_{e^s} and t_{e^o} denote the embedding of the subject s and the object o, respectively; $e^{r^{-1}}$ denotes the embedding of relation r^{-1}, where r^{-1} denotes the inverse of the relation r; and $\langle \cdot \rangle$ denotes the dot product.

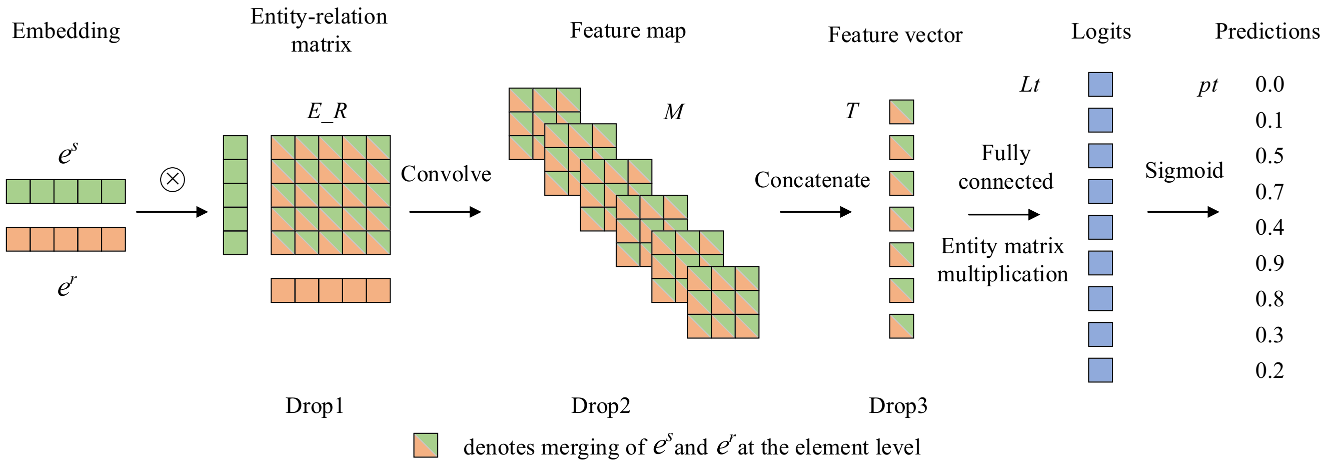

Fig. 2 Design framework of CTKGC

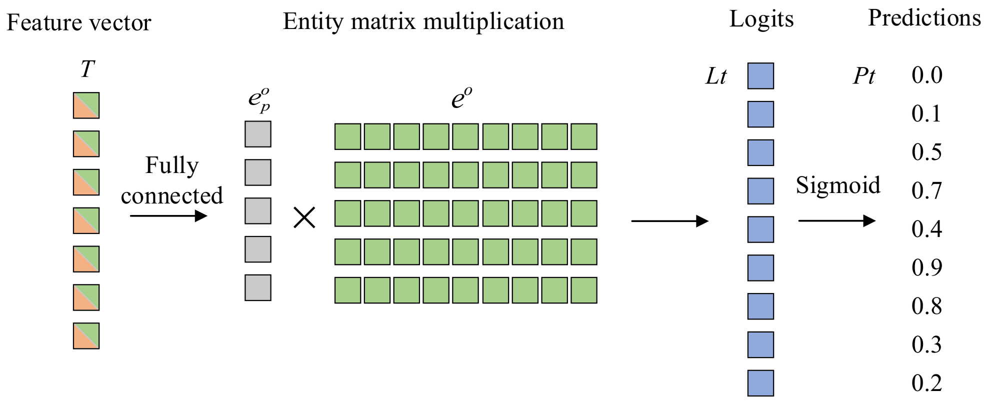

Fig. 3 Framework of the output layer

3 CTKGC model

In this work, the model should use the subject and the relation to predict the object. We propose a mapping relationship of $s + r \approx o$ and utilize embedding to represent the entities and relations of the knowledge graph in a low-dimensional continuous vector space. Then, the interaction between the subject and the relation is modeled via fusion at the element level.

The design framework of the CTKGC model is illustrated in Fig. 2. This model consists of four parts: the embedding, fusion, convolutional, and output layers.

In the embedding layer, the subject and the relation are inputted to obtain the corresponding embedding. In the fusion layer, subject embedding and relation embedding are fused at the element level to obtain the entity−relation matrix. In the convolutional layer, convolution operation is performed on the entity−relation matrix to obtain different feature maps. Then, we transform them into a feature vector through a reshaping operation. In the output layer, the feature vector is matched with all the candidate object embeddings and then scored by the sigmoid function to obtain the score of each candidate object.

3.1 Embedding layer

The facts in a knowledge graph are stored as triples; thus, each element in a triplet is a word or a symbol, such as (Apple Inc., BelongTo, U.S.) in Fig. 1, and cannot be computed using a neural network. Accordingly, we utilize embedding to represent each subject $s \in \{s_1, s_2, s_3, ..., s_{n_e}\}$ and relation $r \in \{r_1, r_2, r_3, ..., r_{n_r}\}$, and generate the corresponding subject embedding $e^s = \{e_1^s, e_2^s, e_3^s, ..., e_{d_e}^s\} \in \mathbb{R}^{n_e \times d_e}$ and relation embedding $e^r = \{e_1^r, e_2^r, e_3^r, ..., e_{d_r}^r\} \in \mathbb{R}^{n_r \times d_r}$, and then enter them into the fusion layer.

3.2 Fusion layer

To integrate subject embedding and relation embedding, and thus, better simulate the object and facilitate 2D convolution operations, we operate $\otimes$ on subject embedding e^s and relation embedding e^r at the element level to obtain the entity−relation matrix $E\_R \in \mathbb{R}^{d_e \times d_r}$. The fusion process is shown in (1):

$$\left(e_1^s, e_2^s, e_3^s, ..., e_{d_e}^s\right)^T \otimes \left(e_1^r, e_2^r, e_3^r, ..., e_{d_r}^r\right) = \begin{vmatrix} e_1^s e_1^r & e_1^s e_2^r & e_1^s e_3^r & \cdots & e_1^s e_{d_r}^r \\ e_2^s e_1^r & e_2^s e_2^r & e_2^s e_3^r & \cdots & e_2^s e_{d_r}^r \\ e_3^s e_1^r & e_3^s e_2^r & e_3^s e_3^r & \cdots & e_3^s e_{d_r}^r \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ e_{d_e}^s e_1^r & e_{d_e}^s e_2^r & e_{d_e}^s e_3^r & \cdots & e_{d_e}^s e_{d_r}^r \end{vmatrix}. \tag{1}$$

From (1), 1D entity embedding and relation embedding are transformed into a 2D matrix. Each element in the matrix is generated by multiplying the corresponding elements of subject embedding and relation embedding, reflecting the full integration between embeddings. In addition, each adjacent part of the matrix can represent the local features of entities and relations, making the next

Table 2 Statistics of four datasets

Dataset	#Ent	#Rel	#Triples in training/validation/test		
WN18RR	40,943	11	86,835	3,034	3,134
FB15k-237	14,541	237	272,115	17,535	20,466
KINSHIP	104	25	8,544	1,068	1,074
UMLS	135	46	5,216	652	661

#Ent denotes the number of entities. #Rel denotes the number of relations

Table 3 Knowledge graph completion results for WN18RR and FB15k-237

Model	WN18RR				FB15k-237			
	MRR	Hits@10	Hits@3	Hits@1	MRR	Hits@10	Hits@3	Hits@1
TransE [3][&]	0.226	0.501	–	–	0.294	0.465	–	–
DistMult [22][∗]	0.43	0.49	0.44	0.39	0.241	0.419	0.263	0.155
ComplEx [21][∗]	0.44	0.51	0.46	0.41	0.247	0.428	0.275	0.158
R-GCN [16]	–	–	–	–	0.248	0.417	0.264	0.151
ConvE [5]	0.43	0.52	0.44	0.40	0.325	0.501	0.356	0.237
CrossE [23]	–	–	–	–	0.299	0.474	0.331	0.211
A2N [2]	0.45	0.51	0.46	0.42	0.317	0.486	0.348	0.232
CTKGC(proposed)	**0.459**	**0.521**	**0.472**	**0.426**	**0.340**	**0.517**	**0.372**	**0.252**

2D convolution more meaningful. Then, CNN can be used to perform convolution operations on the entity−relation matrix, and different features can be extracted to fit the final predicted object.

3.3 Convolutional layer

In the convolutional layer, convolution operations are performed on the entity−relation matrix to obtain more local features. Feature maps $M \in \mathbb{R}^{C\times H\times U}$ are obtained through convolution, where C denotes the number of convolution kernels, H and U denote the two dimensions of feature maps, and M is converted into a 1D vector T with length $(C \times H \times U)$ through concatenation. Then, T is inputted into the output layer. The calculation process is shown in (2) and (3).

$$M = E\_R * w, \tag{2}$$
$$T = vec(M), \tag{3}$$

where w denotes the convolution kernel for extracting information from E_R, and $vec()$ denotes the mapping function.

3.4 Output layer

The framework of the output layer is presented in Fig. 3. CTKGC uses a 1-N scoring function (N=n_e) to obtain the scores of all the candidate objects $o \in \mathbb{R}^{n_e}$ and select the object with the highest score as the final prediction result. In the output layer, feature vector T is projected onto the candidate objects' dimension d_e through full connection, and the predicted object embedding e_p^o is generated. The similarity of e_p^o and all the candidate object embeddings e^o is calculated to obtain the logits Lt of each candidate object. We use a sigmoid function to calculate the score of each candidate triple to find the candidate object with the highest score. The calculation process is shown in (4) and (5):

$$Lt = e_p^o \times (e^o)^T, \tag{4}$$
$$Pt = \sigma(Lt), \tag{5}$$
$$s * r = E\_R \approx T \approx e_p^o \approx o, \tag{6}$$

where Lt denotes the logits of each candidate object, $\times$ denotes matrix multiplication, σ denotes the sigmoid function, and Pt denotes the score of each candidate triple. The process of our algorithm is presented in (6). After multiple iterations of the model, the predicted object embedding e_p^o becomes closer to the true object embedding o.

A score function ψ_r, which gives scores to all the candidate triples, can be obtained through the aforementioned processes. Valid triples will receive higher scores than invalid triples. The scoring function of CTKGC is defined as follows:

$$\psi_r(e^s, e^o) = f(vec([(e^s)^T \otimes e^r] * w))\mathrm{W})e^o, \tag{7}$$

where $[(e^s)^T \otimes e^r]$ denotes the merged entity−relation matrix, f denotes a nonlinear function, and $vec()$ denotes the mapping function projected onto d_e-dimensional space using a linear transformation parameterized by the matrix $\mathrm{W} \in \mathbb{R}^{CHU\times d_e}$ and matched with the object embedding e^o via an inner product.

3.5 Loss function optimization

To achieve the best results, we optimize the loss function of CTKGC, which is a binary cross-entropy loss function:

$$L(f, y) = -\frac{1}{n_e}\sum_{(i=1)}^{n_e}\left(y^{(i)}\log\left(Pt^{(i)}\right) + \left(1-y^{(i)}\right)\log\left(1-Pt^{(i)}\right)\right), \tag{8}$$

where n_e denotes the number of candidate objects in a knowledge graph, Pt denotes the score of the predicted triples, and y denotes the binary label vector.

3.6 Algorithm design

Algorithm 1 summarizes the training process of the proposed model. The goal of training the model is to minimize $L(f, y)$.

Algorithm 1 Pseudocode for CTKGC model.

```
Input :Number of entities $n_e$; number of relations $n_r$; number of entity embeddings $d_e$;
number of relation embeddings $d_r$; number of iterations $total\_iterations$;
Output:Triple scores: $Pt$;
1:Initialize entity embedding set $E$, relation embedding set $R$;
2:for num in $total\_iterations$ do:
3:  for triple$(s, r, o) \in G$ do:
4:    Entity embedding: $e^s \in E$; relation embedding: $e^r \in R$;
5:    $E\_R \leftarrow$ fusion $e^s$ and $e^r$ at the element level based on Eq. (1);
6:    Perform convolution operations on $E\_R$ based on Eq. (2);
7:    $M \leftarrow$ aggregate all the obtained feature maps;
8:    $T \leftarrow$ convert feature maps $M$ to 1D vector after convolution on the basis of Eq. (3);
9:    $e_p^o \leftarrow$ project $T$ onto the candidate object's dimension to obtain the predicted object embedding;
10:   $Lt \leftarrow$ calculate the similarity of $e_p^o$ and all the candidate object embeddings $e^o$ on the basis of Eq. (4);
11:   $Pt \leftarrow$ use the sigmoid function to calculate the score of $Lt$ on the basis of Eq. (5);
12: end for
13: Iteratively update trainable parameters by minimizing loss $L(f, y)$ on the basis of Eq. (7);
14:end for
```

4 Experiment

4.1 Knowledge graph completion datasets

We use four standard knowledge graph completion datasets, namely, WN18RR, FB15k-237, KINSHIP, and UMLS, to evaluate CTKGC. WN18RR and FB15k-237 are subsets of WN18 and FB15k [3], respectively. Reference [20] pointed out that WN18 and FB15k are easy to deduce because the relation in the training set is leaked; that is, the triple data in the test set can be obtained by flipping the triple data in the training set. Thus, simple models can achieve good effects by using this rule. However, these models cannot obtain good results in other practical problems. By improving the problems of WN18 and FB15k, the inverse of many relations in the training set are deleted from the validation and test sets. Hence, WN18RR and FB15k-237 are derived. KINSHIP contains a set of triples explaining the kinship relationships among members of the Alyawarra tribe from Central Australia [11]. UMLS is from the biomedicine field. The entities are biomedical concepts (e.g., diseases), and the relations are diagnoses and treatments [4]. The specific statistical information is provided in Table 2.

4.2 Experimental setup

We utilize a grid search algorithm for selecting hyperparameters. The default dimension of the subject embedding d_e=200, the default dimension of the relation embedding d_r=200, the learning rate Lr $\in$ {0.001, 0.003, 0.005}, the label smoothing ratio Ls $\in$ {0, 0.1, 0.2, 0.3}, the default batch_size is 128, the embedding dropout Drop1 $\in$ {0.1, 0.2, 0.3, 0.4, 0.5, 0.7, 0.9}, the feature dropout Drop2 $\in$ {0.1, 0.2, 0.3, 0.4, 0.5, 0.6}, the hidden layer dropout Drop3 $\in$ {0.1, 0.2, 0.3, 0.4, 0.5}, and the default size of a convolution kernel is 3$*$200.

For WN18RR, Lr=0.003, Drop1=0.2, Drop2=0.2, and Drop3=0.5 are more conducive to the optimal performance of the model. For FB15k-237, the optimal parameters of CTKGC are Lr=0.003, Drop1=0.2, Drop2=0.4, and Drop3=0.5. For KINSHIP, the optimal parameters are Lr=0.003, Drop1=0.9, Drop2=0.6, and Drop3=0.4. For UMLS, the optimal parameters are Lr=0.003, Drop1=0.9, Drop2=0.2, and Drop3=0.5.

CTKGC performs batch normalization [7] after each operation to prevent overfitting and accelerates convergence. We utilize the Dropout [18] operation to standardize the model in multiple stages, the Adam [10] optimizer to minimize the loss of the model, and label smoothing [19] to reduce the overfitting caused by the saturation of the nonlinear output on the label.

4.3 Evaluation metrics

In this study, the effectiveness of the proposed method is evaluated through four standard evaluation metrics of

Table 4 Knowledge graph completion results for KINSHIP and UMLS

Model	KINSHIP				UMLS			
	MRR	Hits@10	Hits@3	Hits@1	MRR	Hits@10	Hits@3	Hits@1
DistMult [22]	0.685	0.943	0.766	0.553	0.924	0.995	0.962	0.879
ComplEx [21]	0.861	0.977	0.935	0.780	0.944	0.994	0.972	0.914
HypER [1]	0.840	0.982	0.912	0.755	0.894	0.984	0.957	0.822
ConvE [5]	0.83	0.98	0.92	0.74	0.94	0.99	0.96	**0.92**
CTKGC(proposed)	**0.879**	**0.988**	**0.943**	**0.807**	**0.948**	**0.995**	**0.975**	0.915

knowledge graph completion: mean reciprocal rank (MRR), Hits@10, Hits@3, and Hits@1.

$$MRR = \frac{1}{|NS|}\sum_{i=1}^{|NS|}\frac{1}{rank_i}, \tag{9}$$

$$\text{Hits@10} = \frac{|Num10|}{|NS|}, \tag{10}$$

$$\text{Hits@3} = \frac{|Num3|}{|NS|}, \tag{11}$$

$$\text{Hits@1} = \frac{|Num1|}{|NS|}, \tag{12}$$

where $|NS|$ denotes the number of triples, $rank_i$ denotes the ranking of the correct label of the $i-$th triple, $|Num1|$ denotes the number of correct labels in the top one, $|Num3|$ denotes the number of correct labels in the top three, and $|Num10|$ denotes the number of correct labels in the top ten.

MRR is an international universal mechanism for evaluating search algorithms. That is, the score of the first matching result is 1, that of the second matching result is 0.5, and that of the nth matching result is $1/n$. If no matching occurs, then the score is 0. The final score is the average of all the scores, as shown in (9). Hits@k refers to the probability that the correct entity appears in the top k elements, as shown in (10), (11), and (12). Therefore, the higher the values of MRR and Hits@k, the more ideal the experimental results.

During the evaluation, we fix the subject s and utilize all the entities to replace object o to score a particular triple and vice versa, obtaining scores for each combination. These scores are ranked using only the "filtered" setting, i.e., we remove all the real cases except for the current test triple.

4.4 Results and analysis

To verify the effectiveness of the CTKGC proposed in this study, we compare it with existing SOTA methods with high academic influence in related research in recent years. The results are provided in Tables 3 and 4, and the best results are highlighted in bold.

Table 3, [$*$]: Results are taken from [1], [&]: Results are taken from [14]. We reimplement HypER, ComplEx, and DistMult with 200 dimension embeddings and $1-$N scoring for a fair comparison and report the obtained results on UMLS and KINSHIP in Table 4. The other results are from the original papers. Some data in Table 3 are missing because they are not reported in the original papers.

As shown in Table 3, CTKGC achieves the best performance on all the metrics on WN18RR. On Hits@1, CTKGC is 2.6% higher than ConvE and 3.6% higher than DistMult. On FB15k-237, the performance of CTKGC is better than those of the compared models in all the metrics. On MRR, the performance of CTKGC is 1.5% higher than ConvE and 4.6% higher than TransE. On KINSHIP, CTKGC also achieves the best performance on all the metrics, i.e., it is 4.9% higher than ConvE and 3.9% higher than HypER on MRR. On UMLS, CTKGC achieves the best performance on nearly all the indicators, except for Hits@1, wherein ConvE performs better.

The results obtained by CTKGC are not only better than those of other linear models, such as DistMult and ComplEx, but also better than those of many complex deep learning models, such as R-GCN, ConvE, and A2N. The primary reason for such result is that the subject and the relation are more fully integrated into the process of generating a 2D matrix. Thus, each element in the matrix

Table 5 Parameter scaling of ConvE vs. CTKGC

Model	Parameter count	Embedding size	MRR	Hits@10	Hits@3	Hits@1
ConvE	5.05M	200	0.32	0.49	0.35	0.23
ConvE	1.89M	96	0.32	0.49	0.35	0.23
CTKGC	4.30M	200	0.340	0.517	0.372	0.252
CTKGC	1.73M	96	0.338	0.513	0.370	0.251

represents part of the subject and the relation, and local features are more meaningful, making the predicted object closer to the real object. Although ConvE generates a 2D matrix, it reshapes and concatenates subject embedding and relation embedding to change the semantic information. Moreover, each element in the matrix represents part of the subject or the relation. Thus, the fusion of subject embedding and relation embedding via CTKGC will help the model improve its performance and achieve good performance on both datasets. Therefore, CTKGC is highly generalizable and can exhibit better performance in knowledge graph complementation on different datasets.

4.5 Parameter efficiency of CTKGC

The parameter efficiency of ConvE is 17× more efficient than R-GCN and 8× more efficient than DistMult [22]. Simultaneously, ConvE and CTKGC are knowledge graph completion models based on neural networks. For the sake of fairness, CTKGC and ConvE use the same embedding dimension during the experiment. Table 5 provides the parameter scaling of ConvE and CTKGC. The results of ConvE are obtained from [5]. CTKGC has fewer parameters than ConvE under the same dimension embedding, and it achieves better results. When the 1.73 M CTKGC and 1.89 M ConvE are compared, the parameter efficiency of CTKGC is relatively high, i.e., 1.8% higher than ConvE on MRR. When the 1.73 M CTKGC and 4.3 M CTKGC are compared, the performance of the model improves as dimension increases.

5 Discussion

We explore the matrix generation process of ConvE and CTKGC as shown in Fig. 4. For ConvE, entity embedding and relation embedding are divided into multiple fragments and then spliced into a matrix. Each element in the matrix represents either entity information or relation information, and the matrix is only adjacent in the horizontal direction, i.e., the relative position of the original embedding is maintained. In the vertical direction, the relative position of the original embedding is not maintained. Hence, the result of the convolution operation is unrealistic on this matrix.

For CTKGC, we fuse subject embedding and relation embedding at the element level to obtain an entity–relation matrix. Each element in the matrix can represent subject information and relationship information. The matrix is adjacent in both the horizontal and vertical directions. The relative positions of the elements maintain the original positions of the subject and relation embedding elements, which can better represent the semantics of s and r. The information extracted by the convolution operation can better represent the result fused by $s * r$. Therefore, the matrix of merging entities and relations makes CNN operations more meaningful, and the extracted features can better represent the content of the original matrix. Accordingly, the algorithm process of our model is more intuitive and interpretable.

We also explore the influence of different convolution kernel sizes on the knowledge graph completion performance of CTKGC. When the size of a convolution kernel is $p * d_r (p < d_e)$, such as 3∗200, we can extract the information from the entire relation dimension. When the size of a convolution kernel is $d_e * q (q < d_r)$, such as 200∗3, we can extract the information from the entire subject dimension. When the size of a convolution kernel is $p * q (p < d_e, q < d_r)$, such as 3∗3, we can extract local information from the entity–relation matrix. In accordance with the information in Table 6, the model achieves the best results with a convolution kernel size of 3∗200. Although the $p * q$ size convolution kernel can also

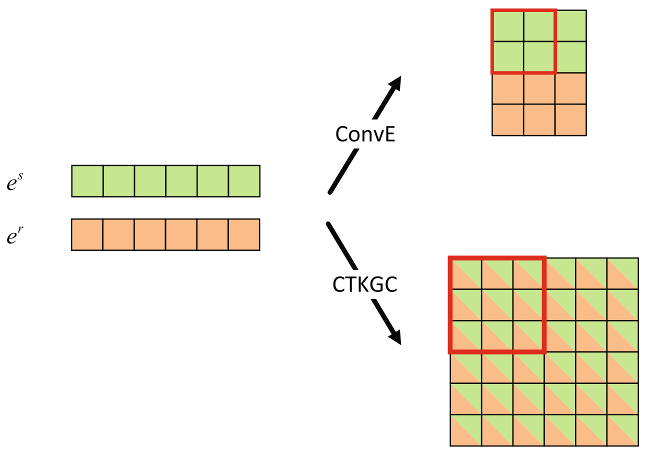

Fig. 4 Matrix generation process

Table 6 Influence of different filter dimension choices on the prediction results

Filter Size	FB15k-237		WN18RR	
	MRR	Hits@1	MRR	Hits@1
3*200	0.340	0.252	0.459	0.426
5*200	0.339	0.251	0.448	0.423
7*200	0.340	0.251	0.450	0.425
200*3	0.338	0.251	0.456	0.424
200*5	0.340	0.251	0.447	0.420
200*7	0.340	0.252	0.449	0.424
3*3	0.336	0.248	0.458	0.429
5*5	0.339	0.249	0.457	0.430

obtain good experimental results, its convolution operation takes a longer time than those of the two other convolution kernel types. The experiments show that CTKGC is insensitive to convolution kernel size, and thus, we recommend using a larger convolution kernel, which requires a shorter time.

6 Conclusions and future work

In this work, we proposed an embedding model, namely, CTKGC, which constructed a new mapping relationship of knowledge graph completion, $s * r \approx o$, and fully integrated subject embedding and relation embedding. Furthermore, we utilized 2D convolution to extract more meaningful local features. The results of CTKGC showed that convolution operation can improve the performance of the model under the full integration of subject and relation semantic information. CTKGC achieved SOTA results on nearly all the metrics on multiple knowledge graph completion datasets.

CTKGC is relatively simple with fewer parameters, and the possibility for development is considerable. It only models subject, relation, and object in a single triple and does not utilize information between different triples. Recent research has found that neighbor node and relation path information in a knowledge graph can play a significant role in inference. Future work may fuse neighbor node and relation path information into the entity−relation matrix to improve the performance of the model.

Acknowledgements This work is supported by the National Natural Science Foundation of China (61602401), Scientific and technological research projects of colleges and universities in Hebei Province (QN2018074), Scientific and technological research projects of colleges and universities in Hebei Province (ZD2019004) and the Nature Scientist Fundation of Hebei Province (F2019203157).

References

1. Balažević I, Allen C, Hospedales TM (2019) Hypernetwork knowledge graph embeddings. In: International conference on artificial neural networks. Springer, pp 553–565
2. Bansal T, Juan DC, Ravi S, McCallum A (2019) A2n: attending to neighbors for knowledge graph inference. In: Proceedings of the 57th annual meeting of the association for computational linguistics, pp 4387–4392
3. Bordes A, Usunier N, Garcia-Duran A, Weston J, Yakhnenko O (2013) Translating embeddings for modeling multi-relational data. In: Advances in neural information processing systems, pp 2787–2795
4. Das R, Dhuliawala S, Zaheer M, Vilnis L, Durugkar I, Krishnamurthy A, Smola A, McCallum A (2017) Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In: International conference on learning representations
5. Dettmers T, Minervini P, Stenetorp P, Riedel S (2018) Convolutional 2d knowledge graph embeddings. In: Proceedings of the AAAI conference on artificial intelligence, vol 32
6. Dong X, Gabrilovich E, Heitz G, Horn W, Lao N, Murphy K, Strohmann T, Sun S, Zhang W (2014) Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, pp 601–610
7. Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proceedings of the 32nd international conference on machine learning, pp 448–456
8. Ji G, He S, Xu L, Liu K, Zhao J (2015) Knowledge graph embedding via dynamic mapping matrix. In: Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers), pp 687–696
9. Kazemi SM, Poole D (2018) Simple embedding for link prediction in knowledge graphs. In: Advances in neural information processing systems, pp 4284–4295
10. Kingma DP, Ba J (2015) Adam: a method for stochastic optimization. In: International conference on learning representations
11. Kok S, Domingos P (2007) Statistical predicate invention. pp 433–440
12. Lin Y, Liu Z, Sun M, Liu Y, Zhu X (2015) Learning entity and relation embeddings for knowledge graph completion
13. Mouromtsev D, Wohlgenannt G, Haase P, Pavlov D, Emelyanov Y, Morozov A (2018) A diagrammatic approach for visual question answering over knowledge graphs. In: European semantic web conference. Springer, pp 34–39
14. Nguyen TD, Nguyen DQ, Phung D et al (2018) A novel embedding model for knowledge base completion based on convolutional neural network. In: Proceedings of the 2018 conference of the North American chapter of the association for computational linguistics: human language technologies, Volume 2 (Short Papers), pp 327–333
15. Nickel M, Tresp V, Kriegel HP (2011) A three-way model for collective learning on multi-relational data. In: Icml, vol 11, pp 809–816
16. Schlichtkrull M, Kipf TN, Bloem P, Van Den Berg R, Titov I, Welling M (2018) Modeling relational data with graph convolutional networks. In: European semantic web conference. Springer, pp 593–607
17. Sheng Y, Lan M (2019) Residual connection-based multi-step reasoning via commonsense knowledge for multiple choice machine reading comprehension. In: International conference on neural information processing. Springer, pp 340–352
18. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958
19. Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z (2016) Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2818–2826
20. Toutanova K, Chen D (2015) Observed versus latent features for knowledge base and text inference. In: Proceedings of the 3rd workshop on continuous vector space models and their compositionality, pp 57–66

21. Trouillon T, Dance CR, Gaussier É, Welbl J, Riedel S, Bouchard G (2017) Knowledge graph completion via complex tensor factorization. J Mach Learn Res 18(1):4735–4772
22. Yang B, Yih SWT, He X, Gao J, Deng L (2015) Embedding entities and relations for learning and inference in knowledge bases. In: Proceedings of the international conference on learning representations
23. Zhang W, Paudel B, Zhang W, Bernstein A, Chen H (2019) Interaction embeddings for prediction and explanation in knowledge graphs. In: Proceedings of the twelfth ACM international conference on web search and data mining, pp 96–104

Jianzhou Feng received M.S. and Ph.D. degrees from Yanshan University, Qinhuangdao, China, in 2007 and 2013, respectively. He is an Associate Professor of Yanshan University. His research interests include natural language processing and knowledge graph.

Qikai Wei is currently pursuing the M.S. degree in Computer Science and Technology from Yanshan university. His research interests include knowledge graph.

Jinman Cui is currently pursuing the M.S. degree in Computer Science and Technology from Yanshan university. Her interests include knowledge graph.

Jing Chen received M.S. and Ph.D. degrees from Yanshan University, Qinhuangdao, China, in 2002 and 2007, respectively. She is an Associate Professor of Yanshan University. Her research interests include peer-to-peer network, social computing, Web service and SNS.