Neural Computing and Applications (2021) 33:11157–11173
https://doi.org/10.1007/s00521-020-05654-4

ORIGINAL ARTICLE

Learning knowledge graph embedding with a bi-directional relation encoding network and a convolutional autoencoder decoding network

Kairong Hu[1] · Hai Liu[1] · Choujun Zhan[1] · Yong Tang[1] · Tianyong Hao[1]

Received: 31 August 2020 / Accepted: 19 December 2020 / Published online: 7 January 2021

Abstract

Derived from knowledge bases, knowledge graphs represent knowledge expressions in graphs, which utilize nodes and edges to denote entities and relations conceptually. Knowledge graph can be described in textual triple form, consisting of head entities, tail entities and relations between entities. In order to represent elements in knowledge graphs, knowledge graph embedding techniques are proposed to map entities and relations into continuous vector spaces as numeric vectors for computational efficiency. Convolution-based knowledge graph embedding models have promising performance for knowledge graph representation learning. However, the input of those neural network-based models is frequently in handmade forms and may suffer from low efficiency in feature extraction procedure of the models. In this paper, a convolutional autoencoder is proposed for knowledge graph representation learning with entity pairs as input, aiming to obtain corresponding hidden relation representation. In addition, a bi-directional relation encoding network is utilized to represent semantic of entities in different directional relation patterns, as an encoder to output representation for initialization of the convolutional autoencoder. Experiments are conducted on standard datasets including, WN18RR, Kinship, NELL-995 and FB15k-237 as a link prediction task. Besides, input embedding matrix composed of different ingredients is designed to evaluate performances of the convolutional autoencoder. The results demonstrate that our model is effective in learning representation from entity feature interactions.

Keywords Bi-directional relation · Convolutional autoencoder · Knowledge graph · Link prediction

1 Introduction

A knowledge graph is composed of numerous "entity-relation-entity" triples that assemble into a linked graph, which are from knowledge bases, such as Freebase [1], WordNet [2] and Yago [3]. Triple form data are a basic element belonging to the set of knowledge items in knowledge graph. Valid triple in knowledge graph consists of head entity, relation and tail entity, in which a relation describes the relationship between a head and a tail entity. Knowledge graph plays an important role in various research topics, such as information extraction, question answering and recommendation system [4]. Driven by learned features of knowledge in knowledge graph, downstream tasks like relation extraction [5], knowledge graph-based reasoning in question answering [6] and knowledge guided recommendation [7] are supported by different kinds of knowledge graphs. To represent knowledge graph for efficient computation, knowledge graph embedding (KGE) techniques are proposed to acquire entity and relation embedding vectors, which are mapped from textual triples (i.e., "entity-relation-entity" pairs) into a continuous vector space [4].

Knowledge graph embedding models leverage elaborate semantic constraints over the embedding vectors of entities

✉ Tianyong Hao
haoty@m.scnu.edu.cn

Kairong Hu
2018022615@m.scnu.edu.cn

Hai Liu
namelh@gmail.com

Choujun Zhan
zchoujun2@gmail.com

Yong Tang
ytang@m.scnu.edu.cn

[1] School of Computer Science, South China Normal University, Guangzhou, China

and relations. As typical translational distance models, TransE [8] was proposed to impose translational distance constraint $\boldsymbol{h} + \boldsymbol{r} \approx \boldsymbol{t}$ for given valid triples. The constraint ensured that the representations of head and tail entities could be explained by their corresponding relations. Different from translational distance models, semantic matching model RESCAL [9] was designed to capture interaction features between head and tail entities by matrices associated with corresponding relations. In recent years, convolution-based neural network models [10–12] have gained attentions for their promising performances in knowledge graph representation learning since the high efficiency and effectiveness of convolutional operations.

Taking node connectivity features into account, entity connectivity features were utilized for knowledge graph representation learning, inspired by graph neural networks, such as GCN [13] and GAT [14]. Graph convolutional network GCN and graph attention network GAT incorporated features of nodes from their neighborhood representation, showing powerful feature extraction on graph. Inspired by GAT, KBAT [15] was proposed to learn entity representation on basis of multi-hop entity representations via an encoder–decoder architecture. BDRAN [16] was designed to capture neighborhood entity features according to different directions of relations in relational patterns. Node connectivity features are considered to yield entity representation through graph attention mechanism in encoding process. However, the input of the decoder models presents simple concatenation of embedding vectors of entities and relations, resulting in limited interactions between entity and relation embedding vectors. Therefore, as an encoder–decoder architecture, this paper utilizes a bi-directional relation encoding network as encoder to capture neighborhood entity features in different relation patterns for initialization. A novel convolution-based autoencoder is proposed as the decoder model, capturing interaction features between two entities as pairs and learning corresponding relation encoding, given textual triples in knowledge graph. As a link prediction task, experiments are conducted on commonly used standard datasets with metrics mean reciprocal ranks (MRR) and the percentage of hits within N (Hits@N). The results show an improvement of our model with 1.8% of MRR, 3.0% of Hits@1 and 1.5% of Hits@3 on average. Categorized relations of triples in the datasets are further used for experiments to demonstrate the capability of our model on complicated relation patterns prediction. In addition, comparative experiments showed effectiveness of our model on capturing entity interaction features by considering the pairs of entity embedding vectors as input of the decoder.

The contribution of this paper mainly lies on following aspects:

- A bi-directional relation encoding network is leveraged to encode entity representation from neighborhood entities through a graph attention mechanism, which is used as initialization for decoder model.
- As an unsupervised method, convolutional autoencoder is proposed to learn hidden relation representations from entity pairs given textual triples for knowledge graph representation learning.
- Evaluation for link prediction task on standard datasets demonstrates improvement of our model comparing with baseline methods. Categorized relations in the datasets are considered to show the effectiveness of our model in predicting entities of complicated relation patterns. In addition, different forms of input are tested to verify the effectiveness of convolutional autoencoder for feature extraction in knowledge graph representation learning.

2 Related work

2.1 Knowledge graph embedding

A knowledge graph can be presented as a graph form or textual form data, consisting of numerous triples as its elements that assemble into a linked graph. Generally, a triple is composed of a head entity, a relation and a tail entity, describing fact in a knowledge graph. In recent years, a knowledge graph representation learning technique called knowledge graph embedding was explored for embedding vector acquisition, which maps text form triples to numeric forms for efficient computation. At the same time, knowledge graph embedding models not only learn entity and relation embedding vectors but also capture features by certain semantic constraints. Inspired by word representations in Mikolov et al. [17], TransE modeled the difference between head and tail entities conformed to translational distance as corresponding relation between them. In other words, the translational distance constraint in TransE was a semantic constraint $\boldsymbol{h} + \boldsymbol{r} \approx \boldsymbol{t}$, on entity vectors and relation vector for a triple (h, r, t) in knowledge graph. However, there existed problems when TransE dealing with 1-to-N, N-to-1 and N-to-N relations. For instance, an embedding vector of a tail entity in 1-to-N relation situation was tended to be learned closer to other tail entity vectors corresponding to the same relation. Thus, 1-to-N, N-to-1 and N-to-N problems might cause inefficiency on representing head or tail entities in KGE models. Therefore, KGE models TransH [18], TransR [19] and TransD [20] were designed to impose translational distance constraints through different entity projection strategies for such problems alleviation. Different from TransE, TransM

[21] constrained the embedding vectors of head (tail) entities and relations to slightly different embeddings of tail (head) entities, depending on different types of relations (i.e., 1-1, 1-N, N-1 and N-N). ManifoldE [22] leveraged a manifold function to impose the embedding vectors of head or tail entities within some manifold spaces. Instead of modeling entities and relations as real vectors, RotatE [23] considered the embedding vectors of relations as rotations from source entities to target entities in a complex space.

In addition, tensor factorization model RESCAL [9] simulated interactions between entities by matrices corresponding to specific relations, which differed from translational distance models. RESCAL required more parameters by capturing interactions between head and tail entities with a matrix, which associated with relation. Therefore, vector forms of relations were introduced in DistMult [24], so as to decrease model parameters through restricting the interaction matrices to diagonal matrices. In order to increase the interaction between head and tail entities, circular correlation operation was used to design the score function in an expressive model HolE [25]. ComplEx [26] extended the representations of entities and relations by using embedding vectors in a complex space. An expressive knowledge graph embedding model named SimplE [27] used two vectors for each entity to learn independent parameters through simplifying ComplEx by removing redundant computation. Based on Tucker decomposition, TuckER [28] represented interactions between a head entity, a relation and a tail entity as an element in core tensor. The aforementioned semantic matching models, such as RESCAL, DistMult, ComplEx and SimplE, can be viewed as special cases of TuckER. In addition, utilizing one of the translational distance models or semantic matching models as a generator or a discriminator, some generative adversarial frameworks (e.g., KBGAN [29], IGAN [30], KSGAN [31] and NKSGAN [32]) were proposed to boost the performance of discriminator, which was trained with positive samples and the high-quality negative samples provided by generator. In recent years, inspired by convolution operation, convolution-based knowledge graph embedding models, such as ConvE [10], ConvKB [11] and CapsE [12], designed different strategies to capture features between entities and relations for knowledge graph representation learning. These promising methods attracted much attentions because of the high effectiveness and efficiency of convolution operation in representation learning.

However, the aforementioned models did not explicitly leverage the features of local structure of nodes, which might contribute to representation in graphs. Node connectivity features were used to aggregate for entity representation generation in R-GCN [33], which was extended from GCN [13]. R-GCN was a relational graph encoder that encoded neighborhood entity features and self-loop representation iteratively, which could be used for entity classification and link prediction with other knowledge graph embedding models. Besides, different from the aggregation strategy in graph convolutional networks, attention importance was computed by graph attention mechanism in GAT [14], according to different nodes and edges representations in local structure. To emphasize distinct representation power of neighborhood entities and the linked edges, different weights were allocated to represent a node when aggregating neighborhood features in GAT. In knowledge graph representation learning, KBAT [15] extended GAT by exploring multi-hop representation of a given entity for representation aggregation via multi-head attention and graph attention mechanism. Inspired by KBAT, BDRAN [16] was designed to represent entities by encoding bi-directional relation features in entity representation level and computing different semantic weights in semantic representation level. Generating various of coarsened graphs, M-GNN [34] learned hierarchical features by a graph coarsening scheme for representation learning in knowledge graph. TransMS [35] utilized a nonlinear function to capture complex multi-directional semantics, which benefited link prediction task in knowledge graph. Following BDRAN, this paper leverages a bi-directional relation encoding network as encoder, so as to capture neighborhood features in different relation patterns for entity representation.

2.2 Autoencoders

In early research, as a multilayer artificial neural network, autoencoder incorporated hidden units between input layer and output layer for internal representation learning by minimizing reconstruction error [36]. Autoencoder was a multilayer network, used to conduct dimensionality reduction of feature vectors in feature space [37, 38]. To obtain representative initialization of deep networks for optimization solution in training procedure, greedy layer-wise pre-training [39] was applied to pre-train stacked autoencoders in which autoencoders acted as building blocks. In addition, there existed variants of traditional autoencoders that learned different forms of representations via different constraints or structures. An extended version of traditional autoencoder, generalized autoencoder [40], was proposed to learn structure relationships between training data and a set of data instances by minimizing a weighted reconstruction error, which was shown as a general framework for dimensionality reduction. Ranzato et al. [41] proposed an energy-based model to learn the parameters of encoder and decoder, which allowed the former to predict code vectors and the latter to yield

sparsified code vectors through minimizing encoding prediction energy and reconstruction energy. From the perspectives of manifold learning and generative model, a denoising autoencoder [42] was introduced to impose autoencoder to learn a mapping function from partial corrupted input data and bring forth robust hidden representations. Such denoising autoencoders could be stacked to yield a deeper network, stacked denoising autoencoder [43], for supervised learning task after pre-training and fine-tuning with an extended unsupervised criterion and corresponding supervised objective, respectively. Encouraging robustness to perturbation of input, contractive autoencoder [44] was developed to explicitly learn contractive mappings by penalizing objective function with a specific regularization, the Frobenius norm of Jacobian matrix. Different from denoising autoencoder, contractive autoencoder directly benefited robustness of extracted features in encoding process. Leveraging a probabilistic encoder network, variational autoencoder [45] reparameterized the sampled random variables (could be viewed as latent representations) from normal distribution and reconstructed the corresponding data with a probabilistic decoder. In computer vision, Ji et al. [46] introduced diverse structures of convolutional neural network-based encoder–decoder model for pixel-wise dense prediction tasks. Proposed by Zhang et al. [47], IntensiveNet consisted of intensive blocks to incorporate features from different layers for text recognition. Convolutional autoencoder [48] and its stacked version utilized convolutional layers and max-pooling layer to preserve shared weights and obtained sparse constraints of latent representations for hierarchical feature extraction in image classification task. As for knowledge graph embedding learning, Takahashi et al. [49] trained a knowledge graph embedding model and a traditional autoencoder jointly, to capture compositional relation constraints for knowledge graph completion task. Inspired by convolutional autoencoder and the joint models mentioned above, this paper proposes a convolution-based knowledge graph embedding model as a decoder to learn entity and relation embeddings which initialized by the output of encoder in a bi-directional relation encoding network.

3 Model

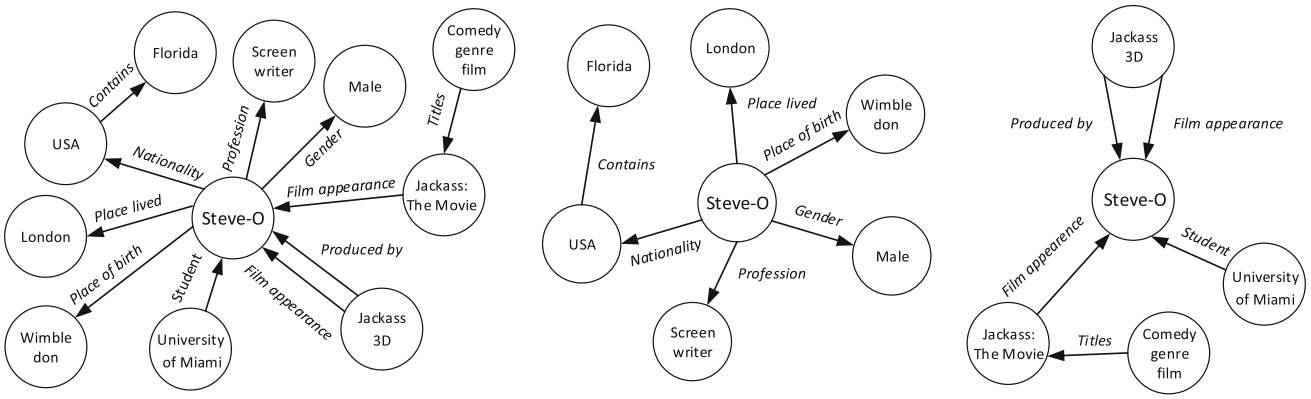
3.1 Notation

Entities (nodes) and relations (edges) connected from head to tail entities are basic ingredients in knowledge graph, in which entities are denoted as e_i and relations are presented as r_k. The subscripts i and k are adopted to refer to a particular entity in entity set $\mathcal{E}$ and a relation in relation set $\mathcal{R}$. Besides, the embedding vectors of entities and relations are denoted as $\boldsymbol{e}_i$ and $\boldsymbol{r}_k$, given an "entity-relation-entity" pair (e_i, r_k, e_j) in a knowledge graph. An entity embraced by its neighborhood entities with in- or out-relations is regarded as a central entity in a subgraph. In- and out-relations of a central entity e_i are presented as $\mathcal{R}^i_{\text{in}} = \{r_k | (e_j, r_k, e_i) \in \mathcal{T}, e_i, e_j \in \mathcal{E}, r_k \in \mathcal{R}\}$ and $\mathcal{R}^i_{\text{out}} = \{r_k | (e_i, r_k, e_j) \in \mathcal{T}, e_i, e_j \in \mathcal{E}, r_k \in \mathcal{R}\}$. The set of neighborhood entities of a central entity e_i is $N(e_i) = \{e_j | (e_i, r_k, e_j) \in \mathcal{T}, e_i, e_j \in \mathcal{E}, r_k \in \mathcal{R}^i_{\text{out}}\} \cup \{e_j | (e_j, r_k, e_i) \in \mathcal{T}, e_i, e_j \in \mathcal{E}, r_k \in \mathcal{R}^i_{\text{in}}\}$, in which a set of triples is denoted as $\mathcal{T}$.

3.2 Encoder

Node connectivity information provides descriptions of nodes in subgraphs, which benefits node representation learning. Graph neural network such as GCN [13] and GAT [14] explored neighborhood representation to encode nodes' connectivity information for effective node representation learning in node classification task. In knowledge graph, relations linked from entities (or a central entity) to a central entity (or other entities), describing different facts of the central entity, which can be utilized for representation. As illustrated in Fig. 1a, a subgraph extracted from the standard dataset FB15k-237, comedian "Steve-O" is a central entity, embraced by neighborhood entities "London" and "University of Miami," and linked by out-relation "Place lived" and in-relation "Student." Consisted of out- or in-relations, Fig. 1b, c is out- and in-relation patterns of Fig. 1a, describing different facts of the central entity. For example, names of location "Wimbledon" and "USA" linked by out-relations "Place lived" and "Nationality" as well as gender description "Male" and its corresponding relation "Gender" give a hint that suggests the central entity "Steve-O" is probably a man with multiple nationalities. In addition, the in-relations connected from neighborhood entities (head entities) to the central entity in Fig. 1c imply that "Steve-O" used to be a student of "University of Miami" and he is now an actor of "Jackass" movie series. It is worth noting that some *2*-hop entities (e.g., entity "Comedy genre film" in the subgraph of in-relations) may provide auxiliary information when representing central entities [15]. From the aforementioned example, elements like entities and relations in an independent subgraph (out-relation or in-relation subgraph) may present different contributions when representing such semantic meanings, while different semantic meanings may have diverse weights when representing a central entity in knowledge graph.

Thus, following BDRAN [16], a bi-directional relation encoding network is leveraged to generate entity representations, which are used for decoder model initialization.



(a) A local relational graph consists of central entity "Steve-O" and neighbor entities.

(b) Out-relations link from head entity "Steve-O" to neighbor entities.

(c) In-relations link from neighbor entities to tail entity "Steve-O".

Fig. 1 Derived from FB15k-237 dataset, **a** presents a local relational graph of central entity "Steve-O," consisting of the central entity and neighborhood entities linked by relations. **b** and **c** are parts of (**a**), depicting out-relation and in-relation patterns. Relations in different directional patterns reflect diverse semantic meanings that can be used to aggregate representation of a given central entity. This paper takes *1*-hop relations and *2*-hop relations into account, e.g., "Student," "Nationality," out–out relation "Nationality"-"Contains" and in–in relation "Titles"-"Film appearance"

There exist two representation levels in general, including an entity representation level and a semantic representation level. In the entity representation level, graph attention mechanism is employed to allocate attention weights to measure representation contribution of neighborhood entities linked by relations. Attention weights are used to aggregate neighborhood features and generate the central entity representation in out-relation pattern or in-relation pattern. In the semantic representation level, central entity representation generated in in-relation or out-relation pattern can be regarded as the representation that expresses specific semantic meanings of the central entity. Multilayer perceptrons (MLPs) are exploited to calculate semantic weights of different semantic meanings to represent an entity in out-relation and in-relation patterns.

Firstly, graph attention mechanism is leveraged to assign attention weights according to the importance of triple hidden representations, to bring forth central entity representations. Hence, triple representations are generated with embedding vectors of entities and relations by simple concatenation. Given a central entity e_i, Eqs. (1) and (2) are representations of triples (e_i, r_k, e_j) and (e_j, r_k, e_i) in out-relation and in-relations pattern.

$$\boldsymbol{v}_{ijk}^{\text{out}} = \left[\boldsymbol{e}_i \| \boldsymbol{e}_j \| \boldsymbol{r}_k\right] \tag{1}$$

$$\boldsymbol{v}_{jik}^{\text{in}} = \left[\boldsymbol{e}_j \| \boldsymbol{e}_i \| \boldsymbol{r}_k\right] \tag{2}$$

It is worth noting that the triple representations are slightly different, since triples (e_i, r_k, e_j) and (e_j, r_k, e_i) are in different directional relation patterns with respect to the central entity e_i.

Following KBAT [15], hidden representation of triple (e_i, r_k, e_j) in out-relation subgraph is presented as $\boldsymbol{u}_{ijk}^{\text{out}}$, shown as Eq. (3).

$$\boldsymbol{u}_{ijk}^{\text{out}} = \boldsymbol{W}_1 \boldsymbol{v}_{ijk}^{\text{out}} \tag{3}$$

The attention importance of such triple in out-relation pattern is measured by a graph attention mechanism, which are the same as the situation of in-relation pattern, shown as Eq. (4).

$$a_{ijk}^{\text{out}} = \text{LeakyReLU}\left(\boldsymbol{W}_2 \boldsymbol{u}_{ijk}^{\text{out}}\right) \tag{4}$$

The importance of triple (e_i, r_k, e_j) is denoted as a_{ijk}^{out} in out-relation pattern, while $\boldsymbol{W}_1$, $\boldsymbol{W}_2$ and LeakyReLU present linear transformation matrices and activation function.

Intuitively, given a central entity e_i, interaction impacts are measured by attention weights $\alpha_{ijk}^{\text{out}}$, which are imposed by neighborhood entity e_j and the relation r connected between them under the pattern of out-relations, as Eq. (5),

$$\alpha_{ijk}^{\text{out}} = \frac{\exp\left(a_{ijk}^{\text{out}}\right)}{\sum_{l \in \mathcal{E}_N} \sum_{r \in \mathcal{R}_{il}} \exp\left(a_{ilr}^{\text{out}}\right)} \tag{5}$$

in which the neighborhood entity set of central entity e_i is denoted by $\mathcal{E}_N$, while $\mathcal{R}_{il}$ is a set consisted of out-relations that connect from central entity e_i to its neighbors e_l.

In the entity representation level, attention weights in both out- and in-relation patterns are brought forth for semantic features generation. According to different semantic features, representation of a central entity is generated in semantic representation level. Following transformer [50] and GAT [14], multi-head attention is

leveraged to stabilize the training process and aggregate semantic features of a central entity in different directional relation patterns in this paper. Specifically, aggregated representation of a central entity e_i with a multi-head attention mechanism is expressed as Eq. (6).

$$\boldsymbol{e}_i^{\text{out}} = \Big\Vert_{m=1}^{M} \sigma\left(\sum_{j\in\mathcal{E}_N, k\in\mathcal{R}_{ij}} \alpha_{ijk,m}^{\text{out}} \boldsymbol{u}_{ijk,m}^{\text{out}}\right) \tag{6}$$

$m = 1,2,\ldots,M$ is the number of attention heads and the output embedding $\boldsymbol{e}_i^{\text{out}}$ is concatenated representation whose elements represent semantic features from diverse semantic spaces of attention. The generated representation contains weighted features of triples in out-relations patterns, in which the most representative triple may have major impact on output embedding vector of a central entity. Linear transformation of relation representations is performed to acquire input vectors in a final attention layer, shown in Fig. 2. The output vector of final attention is represented as $\boldsymbol{e}_i^{\text{out}'}$ in Eq. (7), where $\boldsymbol{u}_{ijk}^{\text{out}'}$ is hidden representation of a triple in final attention layer.

$$\boldsymbol{e}_i^{\text{out}'} = \sigma\left(\sum_{j\in\mathcal{E}_N, k\in\mathcal{R}_{ij}} \alpha_{ijk}^{\text{out}'} \boldsymbol{u}_{ijk}^{\text{out}'}\right) \tag{7}$$

However, the output representations $\boldsymbol{e}_i^{\text{out}'}$ and $\boldsymbol{e}_i^{\text{in}'}$ capture neighborhood features of the central entity only under a particular directional relation pattern (e.g., out- or in-relation pattern) in entity representation level. Therefore, in semantic representation level, we propose aggregation steps to generate final embedding vectors of entities that combine diverse semantic meanings of different relation patterns. To measure importance of representations that express different semantic meanings, we use MLPs and a softmax function to assign representation weights of out- and in-relation patterns as Eq. (8). To preserve entities original semantic information [15], the final representation of a central entity is a weighted sum, including original features and semantic features of directional relation patterns, as Eq. (9).

$$\beta_i^{\text{out}} = \frac{\exp\left(MLP\left(\boldsymbol{e}_i^{\text{out}'}\right)\right)}{\exp\left(MLP\left(\boldsymbol{e}_i^{\text{out}'}\right)\right) + \exp\left(MLP\left(\boldsymbol{e}_i^{\text{in}'}\right)\right)} \tag{8}$$

$$\boldsymbol{e}_i' = \beta_i^{\text{out}} \boldsymbol{e}_i^{\text{out}'} + \left(1 - \beta_i^{\text{out}}\right) \boldsymbol{e}_i^{\text{in}'} + \boldsymbol{W}_e \boldsymbol{e}_i \tag{9}$$

in which linear transformation of original entity representation is presented as $\boldsymbol{W}_e \boldsymbol{e}_i$.

Translational distance constraint is used to constrain the final representations of entities and relations as Eq. (10), and objective function is presented as marginal loss function as Eq. (11). The representations of entities and relations learned from encoding network can be viewed as pre-trained representations of decoder that incorporate

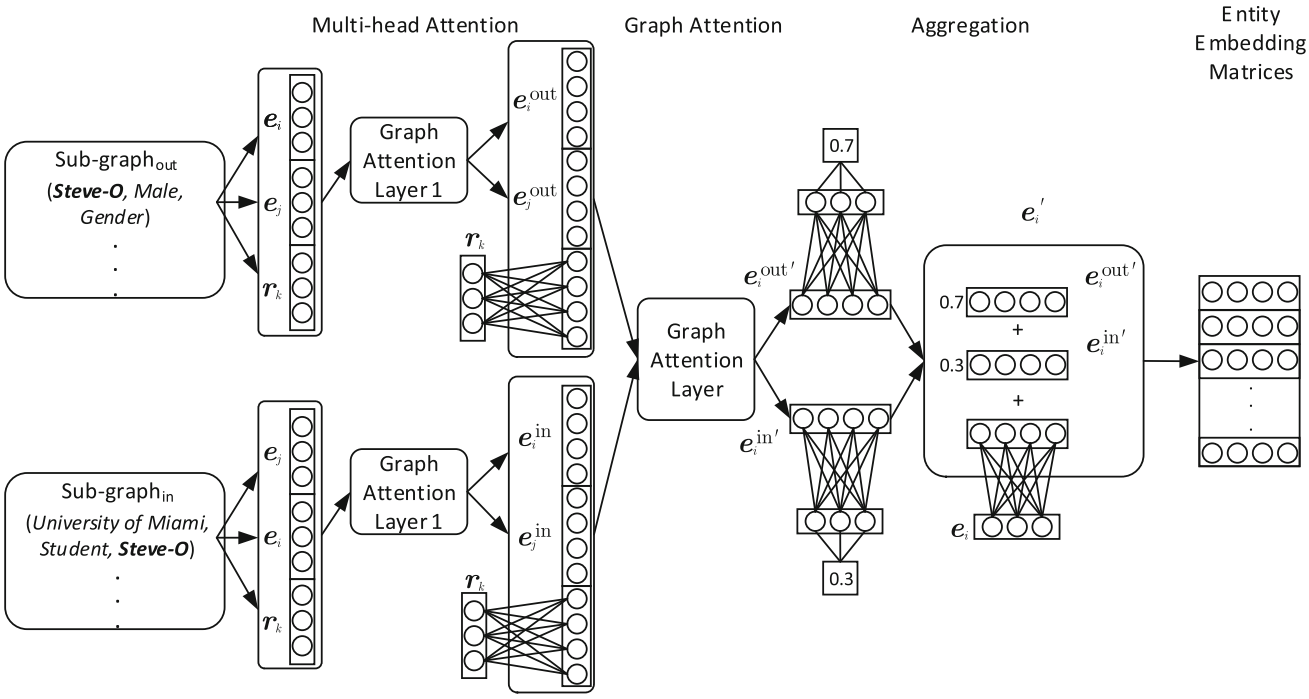


Fig. 2 The overall structure of the bi-directional relation encoding network. As an encoder, graph attention mechanism is leveraged to generate directional relation features of entities in out-relation and in-relation patterns. Distinct semantic information incorporated by the representations in specific directional relation patterns is used to aggregate for final entity representation generation

structural features from entities in subgraphs of knowledge graph,

$$f(e_i, r_k, e_j) = \|\boldsymbol{e}_i + \boldsymbol{r}_k - \boldsymbol{e}_j\| \tag{10}$$

$$L = \sum_{(e_i,r_k,e_j)\in\mathcal{T}} \sum_{(e_i',r_k,e_j')\in\mathcal{T}'} \max\left(0, f(e_i,r_k,e_j) - f(e_i',r_k,e_j') + \gamma\right) \tag{11}$$

in which $\mathcal{T}$ and $\mathcal{T}'$ are sets consisted of positive and negative triples.

3.3 Decoder

As a method of unsupervised learning, autoencoders attempt to recover input data after certain transformations and learn potential representations in hidden space. This paper proposes to capture interaction information between two entity embedding vectors by convolutional autoencoder, as a decoder. As addressed by Gao et al. [51], different regions of an object might have diverse semantic features that needed to be weighted by attention mechanism so as to be interpretable for similarity learning. However, as a decoder in BDRAN, ConvKB treated the embedding vectors of entities and relations as the same type, which may result in poor interpretable and limit the performance of model. Different from model ConvKB, the embeddings of head and tail entity are concatenated to form an input matrix of autoencoder instead of considering the concatenation of the whole triple, as shown in Fig. 3. Since the components of input matrix of convolutional layer are the same types (e.g., entity embeddings), the interaction patterns along every dimension can be captured by convolution kernels effectively, sliding through entity embedding pairs. After convolution procedure, concatenated feature maps are vectorized by concatenation operation and then transformed into latent representation with linear transformation matrix. Aiming to reconstruct original input data, autoencoder adopts transposed operations such as linear transformation with a transposed matrix and transposed convolution. During the training procedure, convolutional autoencoder is imposed to learn hidden representations via constraints of embedding matrices of entity pairs reconstruction. The latent numeric representation can be regarded as the interaction features between these entities in relation feature space. Therefore, the interaction features can be explained as the encoded form of embedding vectors of relations between head and tail entities, which are expected to be similar to the embedding vectors of ground truth relations.

In order to mine relation information between head and tail entities, the input matrix of decoder $\boldsymbol{V} \in \mathbb{R}^{d\times 2}$ is simple concatenation of the embedding vectors of head and tail entity $\boldsymbol{e}_i$ and $\boldsymbol{e}_j$, as shown in Eq. (12).

$$\boldsymbol{V} = \left[\boldsymbol{e}_i^\top \| \boldsymbol{e}_j^\top\right] \tag{12}$$

The i-th output channel of K-channels feature map $\boldsymbol{u} = [\boldsymbol{u}_1\|\boldsymbol{u}_2\|\ldots\|\boldsymbol{u}_K]^\top \in \mathbb{R}^{Kd\times 1}$ in convolutional layer is denoted as d-dimensional feature map $\boldsymbol{u}_i = [u_1, u_2, \ldots, u_d] \in \mathbb{R}^d$, which can be viewed as one of the semantic features extracted by convolution kernel. Equation (13) expresses convolution operation with convolution kernel $\boldsymbol{\omega} \in \mathbb{R}^{1\times 2}$ in the i-th channel.

$$u_i = g\left(\sum_{j=1}^{N} \boldsymbol{V}_{i,j} \cdot \boldsymbol{\omega}_j + b\right) \tag{13}$$

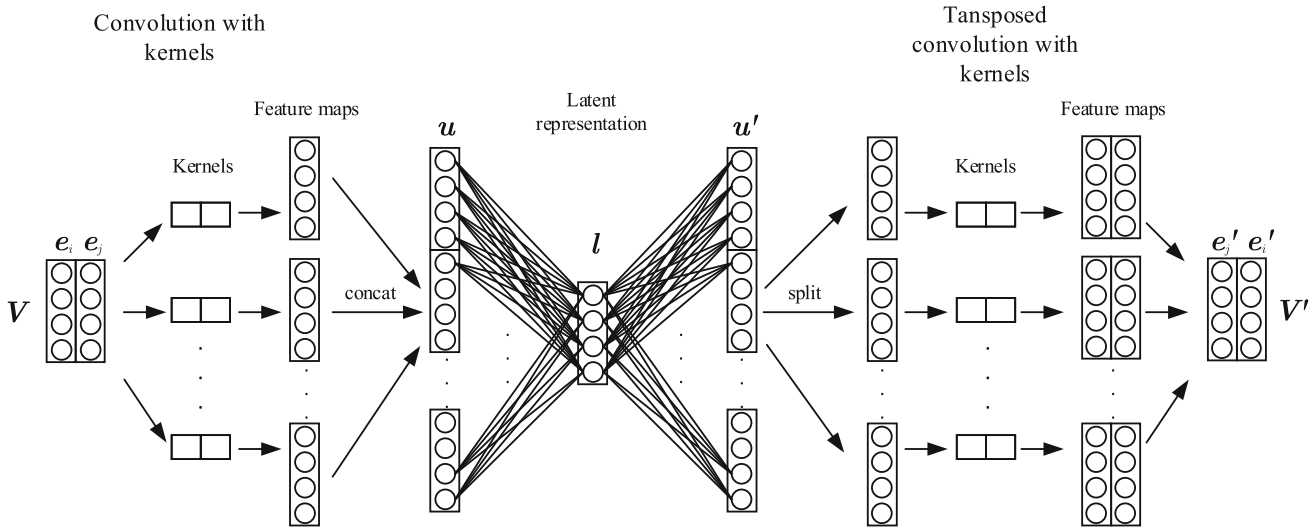


Fig. 3 The proposed convolutional autoencoder for knowledge graph embedding. Entity interactions in entity pairs are captured by filters in convolutional layer, yielding multi-channel feature maps, which are then used to obtain hidden representation. Reconstruction version of entity pair embedding matrix is received after aggregating feature maps by transposed convolution

where N is the number of entity embeddings (e.g., 2) in concatenation, while b and $g(\cdot)$ are the bias and ReLU activation function in the convolutional layer.

Latent representation $\boldsymbol{l} \in \mathbb{R}^d$ (dimensions of both entity and relation embedding vectors are set to d) is learned by projecting multi-channel output of convolutional layer into a relation semantic space with liner transformation matrix $\boldsymbol{W} \in \mathbb{R}^{Kd \times d}$ after convolutional layer, which can be formulated as Eq. (14).

$$\boldsymbol{l} = \boldsymbol{u}^\top \boldsymbol{W} \tag{14}$$

As an embedding vector in relation semantic space, every value of this latent representation $\boldsymbol{l}$ encodes relational interaction information corresponding to every dimension in entity embedding pair. Therefore, similarity score function of given triple (e_i, r_k, e_j) in knowledge graph is shown as Eq. (15), utilizing dot product similarity measurement of embedding vectors.

$$f(e_i, r_k, e_j) = \boldsymbol{l}\boldsymbol{r}_k^\top = \boldsymbol{W}\left(\Big\Vert_{n=1}^{K} g\left(\left[\boldsymbol{e}_i^\top \| \boldsymbol{e}_j^\top\right] * \omega^n\right)\right)\boldsymbol{r}_k^\top \tag{15}$$

where $\Big\Vert_{n=1}^{K}$ denotes concatenation operation over K output feature maps from convolutional layer, while $*$ denotes convolution operation. Latent representation of entity pairs may be similar to corresponding relation embedding vector given valid triple after training process, since the goal of decoder is designed to approximate relation embedding vectors by convolutional autoencoder for given entity pairs.

Training a knowledge graph embedding model with a negative log-likelihood objective function can be regarded as supervised learning given valid triples and corrupted triples as well as their corresponding labels. Knowledge graph learning loss function is shown in Eq. (16). Larger values of scores $f(e_i, r_k, e_j)$ and lower values of scores $f(e_i', r_k, e_j')$ result in lower loss, given valid triples (e_i, r_k, e_j) and corrupted triples (e_i', r_k, e_j').

$$L_1 = \sum_{(e_i, r_k, e_j) \in \mathcal{T} \cup \mathcal{T}'} \log\left[1 + \exp\left(-l \cdot f(e_i, r_k, e_j)\right)\right] \tag{16}$$

Besides, transposed convolution operation [52] and linear transformation with a transposed parameter matrix are leveraged by autoencoder so as to reconstruct input matrix (e.g., the concatenation of entity embedding vectors) after acquiring latent representation. Linear transformation with a transposed matrix $\boldsymbol{W}^\top \in \mathbb{R}^{d \times Kd}$ (tied weights of matrix $\boldsymbol{W}$) is shown as Eq. (17).

$$\boldsymbol{u}' = \boldsymbol{W}^\top \boldsymbol{l}^\top \tag{17}$$

Concatenated feature map $\boldsymbol{u}' \in \mathbb{R}^{Kd \times 1}$ is split into K channels $\boldsymbol{u}_i' \in \mathbb{R}^d, i = 1, \ldots, K$. (Each is d-dimensional feature map, consisting of u_i' as element in every dimension.) Transposed convolution operation on a feature map in one channel n is expressed as follows:

$$\boldsymbol{V}_{i,j}'^n = u_i' \cdot \boldsymbol{\omega}_j'^n \tag{18}$$

where $\boldsymbol{V}_{i,j}'^n$ is the output of transposed convolution with the convolution kernel $\boldsymbol{\omega}_j'^n \in \mathbb{R}^{1 \times 2}$ in channel n. The final output of transposed convolutional layer is the sum of feature maps in every channel as Eq. (19).

$$\boldsymbol{V}_{i,j}' = g'\left(\sum_{n=1}^{K} \boldsymbol{V}_{i,j}'^n + b'\right) \tag{19}$$

$g'(\cdot)$ is denoted as LeakyReLU nonlinear activation function, used to reconstruct output matrix which is imposed to approximate input matrix.

Mean squared error is utilized to measure the error between input matrix of entity embedding vectors $\boldsymbol{V}$ and corresponding output matrix $\boldsymbol{V}'$ as reconstruction loss L_2.

$$L_2 = \frac{1}{|\mathcal{T}_b|}\sum_{n=1}^{|\mathcal{T}_b|}\left(\boldsymbol{V}^{(n)} - \boldsymbol{V}'^{(n)}\right)^2 = \frac{1}{|\mathcal{T}_b|}\sum_{n=1}^{|\mathcal{T}_b|}\sum_{i=1}^{d}\sum_{j=1}^{N}\left(\boldsymbol{V}_{i,j}^{(n)} - \boldsymbol{V}_{i,j}'^{(n)}\right)^2 \tag{20}$$

where $\mathcal{T}_b$ denotes as a set of triples in a batch of training data, including a batch of valid triples and a number of corrupted triples. (The number of corrupted triples is proportional to the number of valid triples.) The loss function of decoder is composed of knowledge graph learning loss and reconstruction loss with L_2 regularization, shown in Eq. 21.

$$L = L_1 + L_2 + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 \tag{21}$$

where $\boldsymbol{w}$ are the embedding parameters of decoder model.

The testing procedure of link prediction task is a ranking process. All the tail (or head) entities can be viewed as candidate tail (or head) entities, to obtain the true answer of a given link prediction task $(e_i, r_k, ?)$ (or $(?, r_k, e_j)$). Semantic scores of candidate triples are evaluated by score function, and the results are ranked to acquire high ranking entities as testing results, when predicting the true tail or head entity. Therefore, this process can be regarded as semantic scores computation given all possible candidate triples. When testing our model given one of the candidate triples (e_i, r_k, e_j') (or (e_i', r_k, e_j)), an input matrix of convolutional autoencoder consists of embedding vector pair $\boldsymbol{e}_i$ and $\boldsymbol{e}_j'$ (or $\boldsymbol{e}_i'$ and $\boldsymbol{e}_j$). Hidden representation of corresponding relation between the input valid head entity and candidate tail entity (or candidate head entity and valid tail entity) is encoded through convolutional layer, as shown in Fig. 3. Note that the embedding vector of relation indicates the potential relation links from given head entity to tail

entity regardless of whether the two entities are directly linked or not in a knowledge graph. The relation representation predicted by convolutional autoencoder is imposed to matched the given relation representation in a testing triple by dot production similarity measurement, as Eq. 22.

$$p_{(e'_i,r_k,e'_j)} = \sigma\left(\boldsymbol{lr}_k^{\top}\right) \tag{22}$$

where $\sigma(\cdot)$ is a sigmoid activation function and $p_{(e'_i,r_k,e'_j)}$ presents the probability of candidate triple (e_i, r_k, e'_j) or (e'_i, r_k, e_j) being valid ones. Higher probability is expected when the encoded relation embedding vector is similar to relation vector in a testing triple, given the valid head (tail) entity and a candidate tail (head) entity, which is actually the valid entity in the testing triple.

4 Experiments

Widely used standard datasets, including WN18RR, Kinship, FB15k-237 and NELL-995, are used to test our model as a link prediction task by comparing with state-of-the-art baseline methods such as KBAT and BDRAN. Our model is further evaluated on the aforementioned datasets, following the categories of relations (e.g., 1-1, 1-N, N-1 and N-N) defined in TransH. Different types of elements in input matrix are considered for feature extraction in convolutional autoencoder.

4.1 Datasets

Standard datasets including WN18RR, Kinship, FB15k-237 and NELL-995 are applied in the experiments. The mean values of in- and out-degree are relatively large in FB15k-237 and Kinship, compared with WN18RR and NELL-995. The statistics of the four datasets are reported in Tables 1 and 2.

- **WN18RR** is extracted from dataset WN18, a subset of WordNet knowledge base, released by [10]. There existed 11 types of relations in WN18RR, such as "hypernym" and "hyponym" that provide words lexical facts.
- **Kinship** is a dataset providing personal relationship descriptions of Alyawarra tribe in Central Australia.

Table 2 The degree statistics of entities in training sets

Datasets	Mean (in-degree)	Mean (out-degree)
WN18RR	2.72	2.19
Kinship	82.15	82.15
FB15k-237	20.34	19.75
NELL-995	4.21	2.77

- **FB15k-237** is released by [53]. It is derived from FB15k which is a subset of Freebase knowledge base. FB15k-237 is composed of triples of facts about actors, movies, locations, etc. Compared with FB15k, redundant triples in FB15k-237 are removed, which results in difficulty to learn entity and relation representations.
- **NELL-995** is derived from NELL [54] by [55], providing descriptions of agricultural products, books, animals, etc.

4.2 Baselines

The following baseline methods are used to compare with the performance of our method on the aforementioned standard datasets in the link prediction task.

- **TransE** is a translational distance model proposed by [8]. It applies translational distance constraint on embedding vectors of entities and relations.
- **DistMult** [24] is a semantic matching model designed to capture interaction features between entities by relation matrices.
- **ComplEx** [26] is introduced to represent entities and relations by embedding vectors in a complex space.
- **ConvKB** [11] is a convolutional KGE model in which different strategies are developed to capture features between entities and relations, comparing with ConvE.
- **ConvE** is designed by [10], which is a convolution-based model in knowledge graph representation learning, leveraging 2D convolution operation to learn representations of entities and relations.
- **R-GCN** [33] encodes local structural features of entities through graph convolutional network for link prediction task.

Table 1 The statistics of the used datasets

Datasets	#Entities	#Relations	#Training	#Validation	#Testing
WN18RR	40,943	11	86,835	3034	3134
Kinship	104	25	8544	1068	1074
FB15k-237	14,541	237	272,115	17,535	20,466
NELL-995	75,492	200	149,678	543	3992

- **KBAT** [15] is proposed to capture multi-hop neighborhood features of entities by utilizing a graph attention mechanism.
- **BDRAN** [16] is a knowledge graph representation learning method, leveraging a bi-directional relation encoding network to capture local structural features of entities.

4.3 Evaluation metrics

Mean reciprocal rank (MRR) and Hits@N are frequently used evaluation metrics of link prediction task in knowledge graph, as Eqs. 23 and 24. Applying the filtered settings in [8], the rank of a head or tail entity in a testing triple (e_i, r_k, e_j) is computed within a filtered entity set (i.e., filtered entity set is the set that may generate valid triples without valid head or tail entities in training set). Large value of MRR indicates knowledge graph embedding models have capability on precise entity representations, while Hits@N denotes a rate of head and tail entities that rank within N (1, 3 or 10, etc.).

$$MRR = \frac{1}{2*|\mathcal{T}_t|} \sum_{(e_i,r_k,e_j)\in\mathcal{T}_t} \frac{1}{rank_{e_i}} + \frac{1}{rank_{e_j}} \tag{23}$$

$$Hits@N = \frac{1}{2*|\mathcal{T}_t|} \sum_{(e_i,r_k,e_j)\in\mathcal{T}_t} I(rank_{e_i} \le N) + I(rank_{e_j} \le N) \tag{24}$$

where $|\mathcal{T}_t|$ is the size of testing triple set $\mathcal{T}_t$ and $I(\cdot)$ is an indicator function, while $rank_{e_i}$ and $rank_{e_j}$ denote values of rank for a head entity e_i and a tail entity e_j during prediction.

4.4 Results

Following the settings in TransH, the negative samples are generated with Bernoulli distribution, according to different numbers of head and tail entities corresponding to the relations connected between them. Therefore, the ratios of positive samples and negative samples are tuned in the range of $\{50, 100, 150, 200, 250\}$. In our experiments, the bi-directional relation encoding network is trained 3600 epochs on WN18RR and 3000 epochs for the rest of datasets, with 128 positive triples and negative triples with a corresponding ratio in number, as mini-batches in training set. Multi-head attention mechanism is leveraged, and the number of multi-head is set to 2. The dimensions of output embedding of entities and relations in encoder are set to 200, which is the same as the dimensions of representation in decoder. Convolutional autoencoder is trained 300 epochs with 128 positive samples and their negative samples as mini-batches. The convolution kernels of convolutional autoencoder are presented as 50, 15, 25 and 20 for datasets Kinship, WN18RR, FB15k-237 and NELL-995, respectively.

The results of link prediction experiments are presented in Tables 3 and 4. The performances of our model BDR+CA on the datasets WN18RR, FB15k-237 and Kinship outperform BDRAN on evaluation metrics MRR and Hits@1. However, the performances on Hits@3 and Hits@10 decrease on FB15k-237 and WN18RR. On dataset NELL-995, the performance of our model is slightly worse than BDRAN on MRR, while the result on Hits@1 presents large value of descent. But the results on Hits@3 and Hits@10 reveal larger proportion of predicted entities within top 10.

To show details of the performances of our model, Figs. 4 and 5 are depicted, as ranking distribution of predicted entities by the model on the four datasets. As illustrated in Figs. 4 and 5, the number of first ranked head and tail entities predicted by our model increases on datasets FB15k-237, Kinship and WN18RR, comparing with baseline model BDRAN. However, the number of entities in low ranking increases on the aforementioned datasets expect for Kinship. As for the dataset NELL-995, the performance of our model in predicting valid head entities (ranked No.1) improves but degrades in predicting valid tail entities, indicating that our model is capable to predict more valid head entities instead of tail entities than BDRAN and keeping balance between head and tail entity predictions.

To further test the performance of our model on link prediction task, relation categories are utilized according to the settings in [18], in which relations are categorized into four types, one-to-one (1-1), one-to-many (1-N), many-to-one (N-1) and many-to-many (N-N). The statistics of the relation categories on the four datasets are shown in Table 5, describing the number of triples that contain the relations in the category in testing data. In addition, the performances of our model BDR+CA and baseline method BDRAN on head and tail entity prediction are illustrated in Tables 6, 7, 8 and 9, categorized by the types of relations in testing triples. As for the performance on triples in relation category N-1, the results on both head and tail entity predictions are slightly worse than BDRAN on FB15k-237, while the performances have improvement of head entity prediction on WN18RR and NELL-995 but have degradation on tail entity prediction with the two datasets. There exist improvements on head and tail entity prediction according to the 1-1 category results on WN18RR, but the values of evaluation metrics MRR and Hits@N are smaller in BDR+CA than in BDRAN on datasets FB15k-237 and NELL-995. It is worth noting that the performances of our model BDR+CA outperform on predicting both head and

Table 3 The performances of our model BDR+CA on WN18RR and NELL-995 datasets compared with baselines. KBAT(out) and KBAT(in) are methods that represent entities by capturing neighborhood features in out-relation and in-relation patterns separately

	WN18RR				NELL-995			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
TransE	0.243	4.27	44.1	53.2	0.401	34.4	47.2	50.1
DistMult	0.444	41.2	47.0	50.4	0.485	40.1	52.4	61.0
ComplEx	0.449	40.9	46.9	53.0	0.482	39.9	52.8	60.6
ConvKB	0.265	5.82	44.5	55.8	0.430	37.0	47.0	54.5
ConvE	0.456	41.9	47.0	53.1	0.491	40.3	53.1	61.3
R-GCN	0.123	8.0	13.7	20.7	0.12	8.2	12.6	18.8
KBAT(in)	0.440	36.1	48.3	58.1	**0.530**	44.7	56.4	**69.5**
KBAT(out)	0.446	37.8	46.9	58.1	0.504	40.6	56.6	66.2
BDRAN	0.446	36.8	**49.0**	**58.2**	**0.530**	**45.0**	57.4	68.0
BDR+CA	**0.468**	**43.0**	48.4	54.0	0.521	41.8	**59.6**	68.4

Bolded values denote the best performance of models

Table 4 The performance comparisons of BDR+CA on FB15k-237 and Kinship datasets

	FB15k-237				Kinship			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
TransE	0.279	19.8	37.6	44.1	0.309	0.9	64.3	84.1
DistMult	0.281	19.9	30.1	44.6	0.516	36.7	58.1	86.7
ComplEx	0.278	19.4	29.7	45.0	0.823	73.3	89.9	97.1
ConvKB	0.289	19.8	32.4	47.1	0.614	43.6	75.5	95.3
ConvE	0.312	22.5	34.1	49.7	0.833	73.8	91.7	98.1
R-GCN	0.164	10.0	18.1	30.0	0.109	3.0	8.8	23.9
KBAT(in)	0.518	46.0	54.0	**62.6**	0.904	85.9	94.1	98.0
KBAT(out)	0.480	41.4	50.5	61.1	0.892	85.2	91.9	97.4
BDRAN	0.539	49.4	**55.5**	**62.6**	0.919	89.2	93.2	97.8
BDR+CA	**0.545**	**51.0**	55.4	61.1	**0.973**	**96.5**	**97.7**	**98.7**

Bolded values denote the best performance of models

tail entities linked by the relations belonged to N-N category, comparing with baseline model BDRAN on all the used datasets. Besides, the numbers of triples in N-N category on all the datasets except for NELL-995 present relatively large values, taking large proportion of complex relations existed in knowledge graph.

Different components of input matrix in convolutional autoencoder are taken into account as different input settings of the decoder model in our experiments. In other words, the embedding vectors of a head entity and its corresponding relation of a given valid or corrupted triple are presented as an input matrix of convolutional autoencoder, which is conducted as comparative experiment comparing with the input setting using the embedding vectors of entity–entity pair (head and tail entity) as input matrix of our model. Tail entity representations are learned by the convolutional autoencoder in the former situation, while relation hidden vectors are encoded in the latter situation, given positive and negative triples in training data. The results are shown in Tables 10 and 11, in which BDR+CA(t) and BDR+CA(r) are denoted as the two situations, respectively. The same hyperparameters of the model, such as number and size of the convolution kernels, are used in the experiments. As shown in Tables 10 and 11, BDR+CA(r) has improvements on WN18RR, NELL-995 and Kinship, while the improvements on FB15k-237 are slight on MRR and Hits@1.

5 Discussion

The same as the baseline method BDRAN, our model leverages a bi-directional relation encoder to learn nodes' connectivity features in graph. BDRAN leverages ConvKB

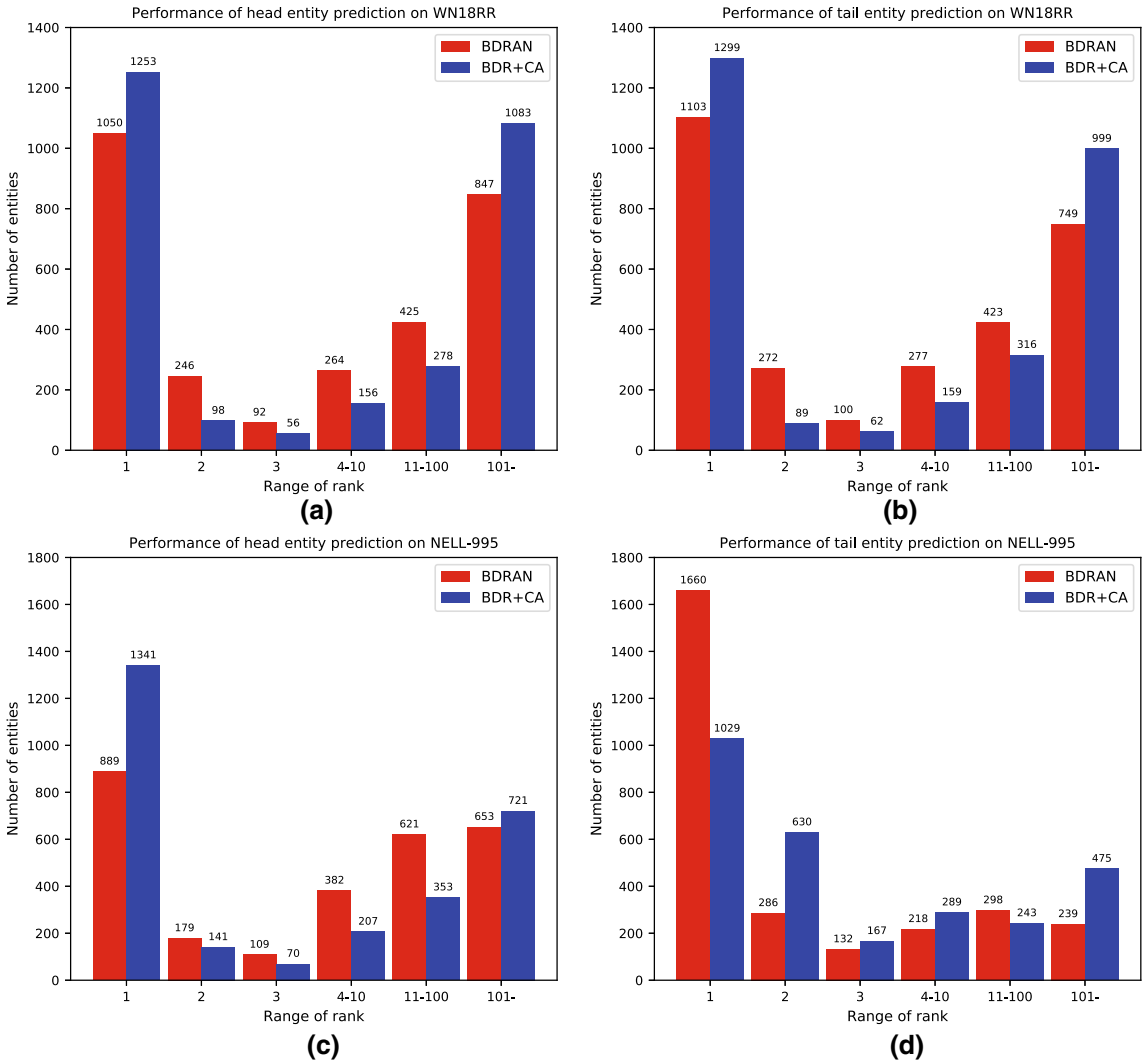


Fig. 4 The figure shows the ranking distribution of head and tail entities predicted by models with testing data of WN18RR and NELL-995. Left bar (red bar) and the right bar (blue bar) present performances of BDRAN and BDR+CA, respectively (color figure online)

as a decoder, while our model considers a convolutional autoencoder. As shown in Tables 3 and 4, the values of our model on MRR and Hits@1 on WN18RR, FB15k-237 and Kinship are relatively large, compared with BDRAN. However, the performances on Hits@3 and Hits@10 decrease on FB15k-237 and WN18RR. The results demonstrate that our model is capable to predict correct head and tail entities of the testing triples, while the correct entities within top 10 remain smaller proportion than baseline BDRAN. Overall, our new proposed decoder convolutional autoencoder outperforms the decoder in BDRAN on predicting entities accurately, demonstrating effectiveness on link prediction. Generally, as illustrated in Figs. 4 and 5, our model is capable to learn representations for correct answer prediction compared with BDRAN, while fails to learn accurate representations of some entities resulting in low ranks of correct ones.

As shown from Tables 6, 7, 8 and 9, the effectiveness of our model on predicting triples in N-N category demonstrates that BDR+CA is capable to model the relations between head and tail entities in knowledge graph and even in circumstance with complex relations by learning hidden representation with convolutional autoencoder.

In the comparative experiments, model BDR+CA has different performances under two input settings, indicating that the input matrix consisted of different elements has impacts on feature extraction with convolution operations. Relation features are extracted by model BDR+CA(r) with multiple convolution kernels, capturing interactions features between head and tail entities effectively along every

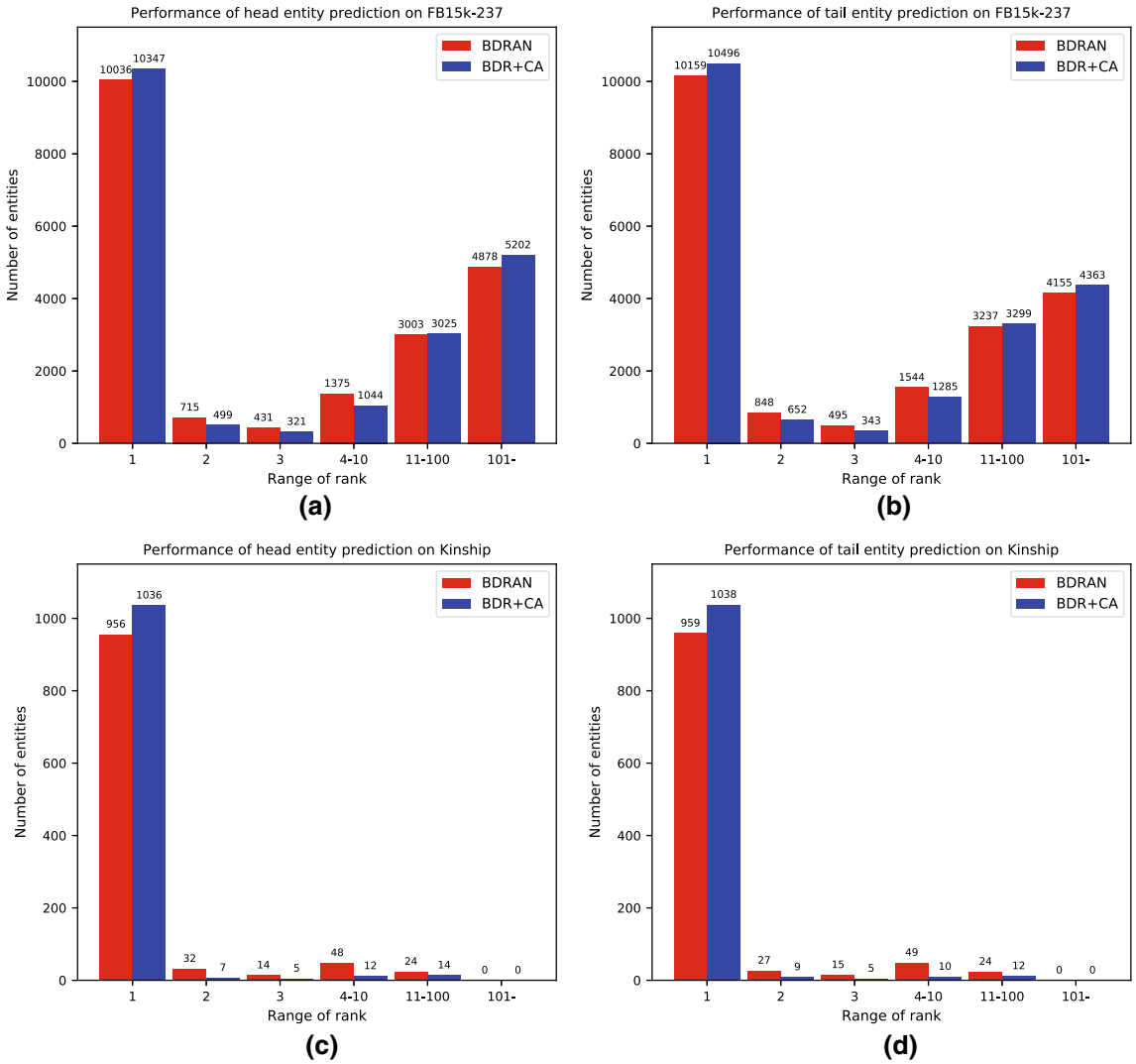


Fig. 5 This figure shows the ranking distribution of head and tail entities predicted by the model with testing data of FB15k-237 and Kinship. In the figure, left bar (red bar) and the right bar (blue bar) present performances of BDRAN and BDR+CA, respectively (color figure online)

Table 5 The statistics of relation categories in testing data of the datasets

Dataset	Category			
	1-1	1-N	N-1	N-N
FB15k-237	192	1290	4490	14,466
Kinship	0	5	0	1069
WN18RR	42	463	1289	1130
NELL-995	178	0	2249	406

dimension of their embedding vectors. Different types of embedding representations may encode diverse semantic features among every dimension of embedding vectors, and convolution operations are shown effectiveness of feature extraction from representations of entity–entity pairs through convolution kernels under current input settings in experiments.

6 Conclusion

This paper leverages a bi-directional relation encoding network as encoder and proposes a novel convolutional autoencoder as decoder for knowledge graph representation learning. As an encoder in encoder–decoder architecture, bi-directional relation encoding network is leveraged to encode bi-directional relation features for entity representation generation and the output representation is used for initialization of decoder. Convolutional autoencoder is trained in unsupervised way to learn hidden representation of relation as a decoder, capturing interaction features between head and tail entities given positive and negative

Table 6 The results of models BDRAN and BDR+CA on dataset WN18RR by relation categories

WN18RR	BDRAN				BDR+CA			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
Head prediction								
1-1	54.0	26.2	85.7	95.2	**92.5**	**88.1**	**97.6**	**97.6**
1-N	**23.4**	**12.3**	**26.8**	**46.2**	16.4	8.6	19.4	32.2
N-1	11.7	5.0	13.7	**24.7**	**14.5**	**10.2**	**16.1**	23.2
N-N	87.3	81.2	93.0	**95.5**	**93.5**	**92.4**	**94.5**	95.0
Tail prediction								
1-1	48.9	23.8	64.3	**97.6**	**95.2**	**92.9**	**97.6**	**97.6**
1-N	**13.9**	**6.9**	**14.9**	**27.2**	6.5	3.0	5.6	13.2
N-1	**22.0**	13.1	**25.8**	**39.3**	21.5	**15.2**	24.5	33.7
N-N	86.0	78.9	92.6	**95.4**	**93.7**	**92.9**	**94.4**	95.0

Bolded values denote the best performance of models

Table 7 The link prediction performance of models BDRAN and BDR+CA on NELL-995 by relation categories

NELL-995	BDRAN				BDR+CA			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
Head prediction								
1-1	**59.4**	**0.50**	**65.7**	**78.7**	57.3	48.9	62.9	73.0
1-N	–	–	–	–	–	–	–	–
N-1	32.8	25.3	34.5	49.5	**49.0**	**44.1**	**50.9**	**58.6**
N-N	64.0	56.9	69.7	75.1	**69.5**	**64.8**	**72.9**	**76.6**
Tail prediction								
1-1	**55.9**	**38.2**	**72.5**	**79.8**	53.7	34.3	**72.5**	78.7
1-N	–	–	–	–	–	–	–	–
N-1	**69.8**	**61.4**	**76.2**	**84.1**	49.5	33.1	63.2	74.0
N-N	56.9	52.0	57.9	64.8	**63.2**	**55.2**	**68.0**	**76.4**

Bolded values denote the best performance of models

Table 8 The performance of models BDRAN and BDR+CA on dataset FB15k-237 by relation categories

FB15k-237	BDRAN				BDR+CA			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
Head prediction								
1-1	**46.4**	**42.7**	**46.4**	**54.7**	42.1	40.1	42.2	45.8
1-N	**32.9**	**26.6**	**34.5**	**44.7**	31.7	25.7	33.3	43.5
N-1	**61.4**	**59.5**	**61.7**	**64.9**	59.0	57.5	59.4	61.7
N-N	52.7	48.0	54.5	**61.9**	**54.3**	**50.9**	**55.2**	60.8
Tail prediction								
1-1	**45.5**	**42.7**	**45.8**	**51.6**	41.1	39.1	40.6	45.8
1-N	**24.5**	**22.5**	**24.9**	**27.7**	21.7	20.2	21.6	24.1
N-1	**68.9**	**64.0**	**71.3**	**78.0**	67.1	62.4	69.3	75.6
N-N	52.9	47.8	54.6	**62.8**	**54.7**	**50.9**	**55.5**	62.1

Bolded values denote the best performance of models

triples in knowledge graph. Experiments are conducted on four commonly used datasets as a link prediction task, comparing with a list of baseline methods. The performances show effectiveness of our model BDR+CA, which is capable to represent complex relations in knowledge graph representation learning. In addition, comparative experiments are presented to show the effectiveness of capturing entity interaction features by convolutional autoencoder.

Table 9 The results of models BDRAN and BDR+CA on Kinship by relation categories

Kinship	BDRAN				BDR+CA			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
Head prediction								
1-1	–	–	–	–	–	–	–	–
1-N	80.6	**0.80**	**0.80**	**0.80**	**81.3**	**0.80**	**0.80**	**0.80**
N-1	–	–	–	–	–	–	–	–
N-N	91.9	89.1	93.4	97.8	**97.2**	**96.5**	**97.7**	**98.8**
Tail prediction								
1-1	–	–	–	–	–	–	–	–
1-N	52.2	0.40	0.60	0.80	**84.0**	**0.80**	**0.80**	**1.00**
N-1	–	–	–	–	–	–	–	–
N-N	92.1	89.5	93.4	97.8	**97.4**	**96.6**	**97.8**	**98.8**

Bolded values denote the best performance of models

Table 10 The results of our model BDR+CA on datasets WN18RR and NELL-995 in comparative experiments. BDR+CA(t) and BDR+CA(r) in the table are the same models with different input elements and learned representation

	WN18RR				NELL-995			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
BDR+CA(t)	34.8	26.6	40.7	49.4	42.2	33.9	48.2	57.4
BDR+CA(r)	**47.3**	**43.6**	**48.9**	**54.2**	**52.1**	**41.8**	**59.6**	**68.4**

Bolded values denote the best performance of models

Table 11 The performances of BDR+CA on datasets FB15k-237 and Kinship

	FB15k-237				Kinship			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
BDR+CA(t)	54.4	50.8	**55.7**	**61.4**	93.1	91.8	93.3	96.4
BDR+CA(r)	**54.5**	**51.0**	55.4	61.1	**97.3**	**96.5**	**97.7**	**98.7**

Bolded values denote the best performance of models

Since feature interaction of the input matrix in decoder model is relevant to the complexity of its organization form, developing elaborated data organization form for input matrix may urge the model to mine meaningful hidden semantic features, which can be considered as future work. In addition, designing a deeper network may be of helpful for effective feature extraction and representation of given well-designed data organization form.

Acknowledgements This work is supported by National Natural Science Foundation of China (No. 61772146, No. 61772211, No. U1811263) and Natural Science Foundation of Guangdong (No. c20140500000225).

Compliance with ethical standards

Conflict of interest The authors confirm that this article content has no conflict of interest.

References

1. Bollacker K, Evans C, Paritosh P, Sturge T, Taylor J (2008) Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of the 2008 ACM SIGMOD international conference on Management of data, pp 1247–1250
2. Miller GA (1995) Wordnet: a lexical database for english. Commun ACM 38(11):39–41
3. Suchanek FM, Kasneci G, Weikum G (2007) Yago: a core of semantic knowledge. In: Proceedings of the 16th international conference on World Wide Web, pp 697–706
4. Wang Q, Mao Z, Wang B, Guo L (2017) Knowledge graph embedding: a survey of approaches and applications. IEEE Trans Knowl Data Eng 29(12):2724–2743
5. Wang G, Zhang W, Wang R, Zhou Y, Chen X, Zhang W, Zhu H, Chen H (2018) Label-free distant supervision for relation extraction via knowledge graph embedding. In: Proceedings of the 2018 conference on empirical methods in natural language processing, pp 2246–2255
6. Zhang Y, Dai H, Kozareva Z, Smola AJ, Song L (2018) Variational reasoning for question answering with knowledge graph. In: Thirty-second AAAI conference on artificial intelligence

7. Xian Y, Fu Z, Muthukrishnan S, De Melo G, Zhang Y (2019) Reinforcement knowledge graph reasoning for explainable recommendation. In: Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval, pp 285–294
8. Bordes A, Usunier N, Garcia-Durán A, Weston J, Yakhnenko O (2013) Translating embeddings for modeling multi-relational data. In: Proceedings of the 26th international conference on neural information processing systems, Volume 2, pp 2787–2795
9. Nickel M, Tresp V, Kriegel H-P (2011) A three-way model for collective learning on multi-relational data. ICML 11:809–816
10. Dettmers T, Minervini P, Stenetorp P, Riedel S (2018) Convolutional 2D knowledge graph embeddings. In: Thirty-second AAAI conference on artificial intelligence
11. Nguyen DQ, Nguyen TD, Nguyen DQ, Phung D (2017) A novel embedding model for knowledge base completion based on convolutional neural network. arXiv preprint arXiv:1712.02121
12. Nguyen DQ, Vu T, Nguyen TD, Nguyen DQ, Phung D (2018) A capsule network-based embedding model for knowledge graph completion and search personalization. arXiv preprint arXiv:1808.04122
13. Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907
14. Veličković P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y (2017) Graph attention networks. arXiv preprint arXiv:1710.10903
15. Nathani D, Chauhan J, Sharma C, Kaul M (2019) Learning attention-based embeddings for relation prediction in knowledge graphs. arXiv preprint arXiv:1906.01195
16. Hu K, Liu H, Zhan C, Tang Y, Hao T (2020) A bi-directional relation aware network for link prediction in knowledge graph. In: International conference on neural computing for advanced applications. Springer, pp 259–271
17. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013) Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems, pp 3111–3119
18. Wang Z, Zhang J, Feng J, Chen Z (2014) Knowledge graph embedding by translating on hyperplanes. In: Twenty-eighth AAAI conference on artificial intelligence
19. Lin Y, Liu Z, Sun M, Liu Y, Zhu X (2015) Learning entity and relation embeddings for knowledge graph completion. In: Twenty-ninth AAAI conference on artificial intelligence
20. 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
21. Fan M, Zhou Q, Chang E, Zheng F (2014) Transition-based knowledge graph embedding with relational mapping properties. In: Proceedings of the 28th Pacific Asia conference on language, information and computing, pp 328–337
22. Xiao H, Huang M, Zhu X (2016) From one point to a manifold: knowledge graph embedding for precise link prediction. In: Proceedings of the twenty-fifth international joint conference on artificial intelligence, pp 1315–1321
23. Sun Z, Deng Z-H, Nie J-Y, Tang J (2018) Rotate: knowledge graph embedding by relational rotation in complex space. In: International conference on learning representations
24. Yang B, Yih W-t, He X, Gao J, Deng L (2014) Embedding entities and relations for learning and inference in knowledge bases. arXiv preprint arXiv:1412.6575
25. Nickel M, Rosasco L, Poggio T (2016) Holographic embeddings of knowledge graphs. In: Proceedings of the thirtieth AAAI conference on artificial intelligence, pp 1955–1961
26. Trouillon T, Welbl J, Riedel S, Gaussier É, Bouchard G (2016) Complex embeddings for simple link prediction. In: International conference on machine learning, pp 2071–2080
27. Kazemi SM, Poole D (2018) Simple embedding for link prediction in knowledge graphs. In: Advances in neural information processing systems, pp 4284–4295
28. Balažević I, Allen C, Hospedales T (2019) Tucker: tensor factorization for knowledge graph completion. In: Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP), pp 5188–5197
29. Cai L, Wang WY (2017) Kbgan: adversarial learning for knowledge graph embeddings. arXiv preprint arXiv:1711.04071
30. Wang P, Li S, Pan R (2018) Incorporating gan for negative sampling in knowledge representation learning. In: Thirty-second AAAI conference on artificial intelligence
31. Hu K, Liu H, Hao T (2019) A knowledge selective adversarial network for link prediction in knowledge graph. In: CCF international conference on natural language processing and Chinese computing. Springer, pp 171–183
32. Liu H, Hu K, Wang F-L, Hao T (2020) Aggregating neighborhood information for negative sampling for knowledge graph embedding. Neural Comput Appl 32:17637–17653
33. 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
34. Wang Z, Ren Z, He C, Zhang P, Hu Y (2019) Robust embedding with multi-level structures for link prediction. In: Proceedings of the 28th international joint conference on artificial intelligence. AAAI Press, pp 5240–5246
35. Yang S, Tian J, Zhang H, Yan J, He H, Jin Y (2019) Transms: knowledge graph embedding for complex relations by multidirectional semantics. In: Proceedings of the 28th international joint conference on artificial intelligence. AAAI Press, pp 1935–1942
36. Rumelhart DE, Hinton GE, Williams RJ (1985) Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science
37. Bourlard H, Kamp Y (1988) Auto-association by multilayer perceptrons and singular value decomposition. Biol Cybern 59(4–5):291–294
38. Hinton GE, Salakhutdinov RR (2006) Reducing the dimensionality of data with neural networks. Science 313(5786):504–507
39. Bengio Y, Lamblin P, Popovici D, Larochelle H (2006) Greedy layer-wise training of deep networks. In: Proceedings of the 19th international conference on neural information processing systems, pp 153–160
40. Wang W, Huang Y, Wang Y, Wang L (2014) Generalized autoencoder: a neural network framework for dimensionality reduction. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pp 490–497
41. Ranzato MA, Poultney C, Chopra S, LeCun Y (2006) Efficient learning of sparse representations with an energy-based model. In: Proceedings of the 19th international conference on neural information processing systems, pp 1137–1144
42. Vincent P, Larochelle H, Bengio Y, Manzagol P-A (2008) Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th international conference on Machine learning, pp 1096–1103
43. Vincent P, Larochelle H, Lajoie I, Bengio Y, Manzagol P-A, Bottou L (2010) Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. J Mach Learn Res 11(12):3371–3408
44. Rifai S, Vincent P, Muller X, Glorot X, Bengio Y (2011) Contractive auto-encoders: explicit invariance during feature extraction. In: Icml

45. Kingma DP, Welling M (2013) Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114
46. Ji Y, Zhang H, Zhang Z, Liu M (2021) Cnn-based encoder-decoder networks for salient object detection: a comprehensive review and recent advances. Inf Sci 546:835–857
47. Zhang Z, Tang Z, Zhang Z, Wang Y, Qin J, Wang M (2020) Fully-convolutional intensive feature flow neural network for text recognition. In: Proceedings of the 24th European conference on artificial intelligence
48. Masci J, Meier U, Cireşan D, Schmidhuber J (2011) Stacked convolutional auto-encoders for hierarchical feature extraction. In: International conference on artificial neural networks. Springer, pp 52–59
49. Takahashi R, Tian R, Inui K (2018) Interpretable and compositional relation learning by joint training with an autoencoder. In: Proceedings of the 56th annual meeting of the association for computational linguistics (volume 1: Long Papers), pp 2148–2159
50. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. In: Advances in neural information processing systems, pp 5998–6008
51. Gao X, Zhang Z, Mu T, Zhang X, Cui C, Wang M (2020) Self-attention driven adversarial similarity learning network. Pattern Recogn 105:107331–107341
52. Dumoulin V, Visin F (2016) A guide to convolution arithmetic for deep learning. arXiv preprint arXiv:1603.07285
53. Toutanova K, Chen D, Pantel P, Poon H, Choudhury P, Gamon M (2015) Representing text for joint embedding of text and knowledge bases. In: Proceedings of the 2015 conference on empirical methods in natural language processing, pp 1499–1509
54. Carlson A, Betteridge J, Kisiel B, Settles B, Hruschka ER, Mitchell TM (2010) Toward an architecture for never-ending language learning. In: Twenty-Fourth AAAI conference on artificial intelligence
55. Xiong W, Hoang T, Wang WY (2017) Deeppath: a reinforcement learning method for knowledge graph reasoning. arXiv preprint arXiv:1707.06690

Terms and Conditions

Springer Nature journal content, brought to you courtesy of Springer Nature Customer Service Center GmbH (“Springer Nature”).
Springer Nature supports a reasonable amount of sharing of research papers by authors, subscribers and authorised users (“Users”), for small-scale personal, non-commercial use provided that all copyright, trade and service marks and other proprietary notices are maintained. By accessing, sharing, receiving or otherwise using the Springer Nature journal content you agree to these terms of use (“Terms”). For these purposes, Springer Nature considers academic use (by researchers and students) to be non-commercial.
These Terms are supplementary and will apply in addition to any applicable website terms and conditions, a relevant site licence or a personal subscription. These Terms will prevail over any conflict or ambiguity with regards to the relevant terms, a site licence or a personal subscription (to the extent of the conflict or ambiguity only). For Creative Commons-licensed articles, the terms of the Creative Commons license used will apply.
We collect and use personal data to provide access to the Springer Nature journal content. We may also use these personal data internally within ResearchGate and Springer Nature and as agreed share it, in an anonymised way, for purposes of tracking, analysis and reporting. We will not otherwise disclose your personal data outside the ResearchGate or the Springer Nature group of companies unless we have your permission as detailed in the Privacy Policy.
While Users may use the Springer Nature journal content for small scale, personal non-commercial use, it is important to note that Users may not:

1. use such content for the purpose of providing other users with access on a regular or large scale basis or as a means to circumvent access control;
2. use such content where to do so would be considered a criminal or statutory offence in any jurisdiction, or gives rise to civil liability, or is otherwise unlawful;
3. falsely or misleadingly imply or suggest endorsement, approval , sponsorship, or association unless explicitly agreed to by Springer Nature in writing;
4. use bots or other automated methods to access the content or redirect messages
5. override any security feature or exclusionary protocol; or
6. share the content in order to create substitute for Springer Nature products or services or a systematic database of Springer Nature journal content.

In line with the restriction against commercial use, Springer Nature does not permit the creation of a product or service that creates revenue, royalties, rent or income from our content or its inclusion as part of a paid for service or for other commercial gain. Springer Nature journal content cannot be used for inter-library loans and librarians may not upload Springer Nature journal content on a large scale into their, or any other, institutional repository.
These terms of use are reviewed regularly and may be amended at any time. Springer Nature is not obligated to publish any information or content on this website and may remove it or features or functionality at our sole discretion, at any time with or without notice. Springer Nature may revoke this licence to you at any time and remove access to any copies of the Springer Nature journal content which have been saved.
To the fullest extent permitted by law, Springer Nature makes no warranties, representations or guarantees to Users, either express or implied with respect to the Springer nature journal content and all parties disclaim and waive any implied warranties or warranties imposed by law, including merchantability or fitness for any particular purpose.
Please note that these rights do not automatically extend to content, data or other material published by Springer Nature that may be licensed from third parties.
If you would like to use or distribute our Springer Nature journal content to a wider audience or on a regular basis or in any other manner not expressly permitted by these Terms, please contact Springer Nature at

onlineservice@springernature.com