Knowledge Base Reasoning with Convolutional-Based Recurrent Neural Networks

Qiannan Zhu, Xiaofei Zhou, Jianlong Tan, and Li Guo

Abstract—Recurrent neural network(RNN) has achieved remarkable performances in complex reasoning on knowledge bases, which usually takes as inputs vector embeddings of relations along a path between an entity pair. However, it is insufficient to extract local correlations of a path due to RNN is better at capturing global sequential information of a path. In this paper, we take full advantages of convolutional neural network that can effectively extract local features, and propose a convolutional-based RNN architecture denoted as C-RNN to perform reasoning. C-RNN first utilizes CNN to extract local high-level correlation features of a path, and then feeds the correlation features into recurrent neural network to model the path representation. Our C-RNN architecture is adaptable to obtain not only local features but also global sequential features of a path. Based on C-RNN architecture, we devise two models, the unidirectional C-RNN and bidirectional C-RNN. We empirically evaluate them on a large-scale FreeBase+ClueWeb prediction task. Experimental results show that C-RNN models achieve state-of-the-art predictive performance.

Index Terms—Knowledge base embedding, knowledge base reasoning, knowledge representation learning, knowledge graph completion

1 Introduction

Recently knowledge bases(KBs) constructed by reasoning on entities and relations have increasingly drawn attention in both industry and academia. Such knowledge bases like FreeBase [1], WordNet [2], YAGO [3], NELL [4] and DBPedia [5] are significant resources for numerous artificial intelligence applications including question answering [6], [7], named entity linking [8]. Although a typical KB contains billions of structured facts(triplets) as (h, r, t) representing that a relation r connects the head entity h to the tail entity t, it is factually far from completion, i.e., plenty of significant facts are missed. Currently many works based on embedding vector representations [9], [10], [11], [12], [13], [14], [15], [16] successfully fill in these missing facts only by one-step[1] paths. Factually more complex reasoning by leveraging multi-hop paths composed of two or more relations in the KB are more helpful for completing the knowledge bases. For instance, Obama-*Nationality*-USA can be inferred by observing the path Obama-*BornInCity*-Honolulu-*CityInState*-Hawaii-*StateInCountry*-USA from the KB.

Previous works about such complex reasoning mainly focused on learning symbolic and logical rules. SHERLOCK system [17] and Path Ranking Algorithm(PRA) [18] both utilize Horn clauses as features in a per-target-relation binary classifier. However such models usually engender massive distinct categories of Horn clauses as distinct features, which restricts the availability of these models to populate large-scale knowledge bases that contain billions of relations. An emerging type of methods based on recurrent neural network (RNN) such as Path-RNN [19] and Single-RNN [20] gains better generalization by reasoning about multi-hop paths. Path-RNN [19] applies RNN to compose relation embeddings along a variable-length path between two entities, inferring new relation embeddings between them. Path-RNN trains a separate model for each relation, and is inaccurate and impractical for large-scale knowledge bases. For enhancing the practicality of Path-RNN, Single-RNN [20] jointly takes as input the relations, entities and entity types, and learns a shared model for all relations by sharing the parameters of relation-specific RNN across all target relations. Although the traditional RNN-based reasoning models are skilled for discovering the sequential information along a path, they have weakness in their ability to capture the local correction within a path. An example is given in Fig. 1. The sequential information of the path Obama-*BornInCity*-Honolulu-*CityInState*-Hawaii-*StateInCountry*-USA is easily to be extracted by RNN, while the local correlation of the path, such as Obama-*BornInCity*-Honolulu-*CityInState*- Hawaii and Honolulu-*CityInState*- Hawaii-*StateInCountry*-USA, is usually ignored for complex path reasoning. Fortunately, convolutional neural network(CNN) has the preeminent capabilities of capturing local features at distinct positions in a sequence through convolutional filters and learning higher-level correlations through pooling operations.

In this paper, we propose a convolutional-based RNN architecture namely C-RNN for complex reasoning over knowledge bases, which takes the superiorities of CNN and RNN. C-RNN constructs a unified architecture by stacking

1. Washington-*LocatedIn*-USA is implied via inferring Washington-*IsCapticalOf*-USA.

• *The authors are with the Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China, and also with the School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China. E-mail: {zhuqiannan, zhouxiaofei, tanjianlong, guoli}@iie.ac.cn.*

Manuscript received 18 Aug. 2018; revised 11 Oct. 2019; accepted 29 Oct. 2019. Date of publication 0 . 0000; date of current version 0 . 0000.
(Corresponding author: Xiaofei Zhou.)
Recommended for acceptance by A. Poulovassilis.
Digital Object Identifier no. 10.1109/TKDE.2019.2951103

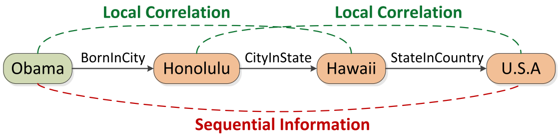

Fig. 1. An example of reasoning on knowledge graph.

the convolutional neural network(CNN) and recurrent neural network(RNN), where the output of one-layer CNN are fed into RNN. Specially, instead of directly applying path embedding to RNN in [19], [20], C-RNN first uses one-layer CNN to extract local higher-level features of the input path embedding, and then rearranges such local features as the input of RNN. Finally for all target relations, C-RNN adds a projection layer to connect the last output vector of RNN. Additionally, we take the entities, entity-types and context of entities into account to provide the richer semantic information for representing the input paths.

Our contributions are as follows:

- We propose a convolutional-based recurrent neural network architecture C-RNN which is the combination of convolutional neural network and recurrent neural network for reasoning on KBs.
- We present seven types of input paths for complex reasoning on KBs, which dynamically generate the entities, entity-types and the context of entities at every KG reasoning step.
- We empirically evaluate our C-RNN models on a large-scale FreeBase+ClueWeb prediction task. Experimental results show that our models achieve improvement on the MAP metric.

We organize the remainder of this paper as follows. In Section 2, we give the preliminaries of the recurrent neural network based reasoning models. Section 3 briefly introduces several basic types of input path. In Section 4, we elaborately describe the framework of C-RNN and devise several unidirectional and bidirectional C-RNN models. For both unidirectional and bidirectional C-RNN, we introduce the separated and shared C-RNN and discuss how such models can be trained on knowledge graphs in Section 5. Section 6 reports the experimental results on FreeBase+ClueWeb data set and undertake some discussions on C-RNN in Section 7. Further we review the applications of CNN (RNN) and the existing related reasoning techniques in Section 8, then followed by the concluding remarks in Section 9.

2 Preliminary

2.1 The Architecture of CNN and RNN

As two widely used neural networks, convolutional neural networks and recurrent neural networks are usually combined and successfully applied in computer vision tasks [21], [22], speech recognition [23] and sentence and document representation [24], [25], sentiment classification [26], [27]. Most of these models train multi-layer CNN and RNN separately or throw the output of a fully connected layer of CNN into RNN as inputs. Our C-RNN architecture is not only different from such above models, but also it is the first work to apply to the knowledge graph research field. More details are illustrated in Section 4.

2.2 RNN for Reasoning

Recurrent neural networks like Path-RNN [19] and Single-RNN [20] models can be seen for complex reasoning over knowledge bases. Both models have the similar basic RNN architecture. Following gives the detailed description:

Path-RNN [19] adopts RNN to reason on an arbitrary-length path between two entities, and non-atomically predicts new relation types between them. Path-RNN gives the representation of path by the last hidden state of RNN obtained after visiting all the relations in the path. Prediction about the new relation types is performed by computing the similarity between the representation of the path and query relation embedding.

Path-RNN defines a path π between an entity pair (e_h, e_t) as $\pi = \{r_1, r_2, \ldots, r_s\} \in S$, and S denotes the set of paths between e_h and e_t. Let $\mathbf{v}(r_t) \in R^d$ be the vector embedding of relation r_t. In step k, the repeated module of Path-RNN takes as input both the vector embedding representing path-so-far $\mathbf{h}_{k-1}$ and the k-step relation vector embedding $\mathbf{v}(r_k)$, and outputs an intermediate embedding representation $\mathbf{h}_k$, given by:

$$\mathbf{h}_k = f(\mathbf{W}_1^r \mathbf{h}_{k-1} + \mathbf{W}_2^r \mathbf{v}^r(r_k)),$$

where $f = sigmoid$, $\mathbf{W}_1^r$ and $\mathbf{W}_2^r$ are the relation-specific parameters of Path-RNN, and r indicates the query relation. Here, every relation that is predicted holds separate parameters $\{\mathbf{v}^r(r_k), \mathbf{W}_1^r, \mathbf{W}_2^r\}$. The last intermediate embedding representation $\mathbf{h}_s$ is the representation of the path $\mathbf{v}(\pi)$. The similarity score is defined as: $score(\pi, r) = \mathbf{v}(\pi) \cdot \mathbf{v}(r)$. Usually there are multiple paths between an entity pair, Path-RNN only selects the max-score path to participates in the query relation r.

Different from Path-RNN, Single-RNN [20] considers entities and defines the path as $\pi = \{e_1, r_1, e_2, r_2, \ldots, r_s, e_t\}$ $(e_h = e_1)$. Further Single-RNN shares the relation embedding representations and the parameters of relation-specific RNN for all target relation types. Thus the hidden state is designed as

$$\mathbf{h}_k = f(\mathbf{W}_1 \mathbf{h}_{k-1} + \mathbf{W}_2 \mathbf{v}(r_k) + \mathbf{W}_3 \mathbf{v}(e_k)).$$

Moreover instead of selecting max-score path, Single-RNN introduces the 'Top-k', 'Average' and 'LogSumExp' scoring pooling methods for deciding which paths can participate in querying relation r. Specifically, 'Top-k' is the average of top-k score paths $\frac{1}{k}\sum_j s_j$, 'Average' is the average of scores of all path $\frac{1}{N}\sum_j^N s_j$, and 'LogSumExp' is $log(\sum_j exp(s_j))$, $s_j \in \{s_1, s_2, \ldots, s_N\}$ is the similarity scores of N paths between an entity pair.

Although Path-RNN and Single-RNN achieve better performances in complex reasoning because of capturing the sequential information of a path, they ignore the local correlations along a path. Inspired by the successful use of the combination of convolutional neural networks and recurrent neural networks for NLP tasks [24], [26], computer vision [21], [22] and speech recognition [23], we will apply CNN to RNN, and devise a novel convolutional-based RNN architecture for complex reasoning over knowledge bases in next section.

3 Input Path

This section defines seven types of input path for our proposed C-RNN architecture. At first, we develop the context

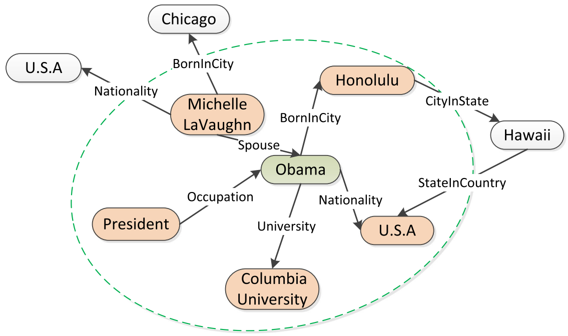

Fig. 2. Illustration of context of an entity in knowledge graph.

information of entities. Usually it is limited that the only learned embedding is used to represent a single entity. In order to identify the position of entities in the path and knowledge base, the additional contextual information for each entity is extracted. We define the "context" of entity e as the set of its first-order neighbors in the knowledge graph, i.e.,

$$context(e) = \{e_i | (e_i, r, e) \in \mathtt{G} \ or \ (e, r, e_i) \in \mathtt{G}\},$$

where r is a relation and $\mathtt{G}$ is a knowledge base. The usage of context could provide more complementary evidence and assist in improving the identifiability of entities. That is because that the contextual entities related to the current entity usually contain the extra semantic and logic information. Here an example of the context information of the entity *Obama* is illustrated in Fig. 2. For the entity *Obama*, we use not only "Obama" itself but also its contexts to represent the entity. The context information of "Obama" are "Honolulu" (BornInCity), "U.S.A"(Nationality), "Michelle LaVaughn" (Spouse), "President"(Occupation), and "Columbia University"(University). There are many works like [28] on using the neighbors-aware context information to learn embeddings. In this paper, we calculate the context embedding as the average of the contextual entities of an entity e

$$\overline{\mathbf{e}} = \frac{1}{|context(e)|} \sum_{e_i \in context(e)} \mathbf{e}_i.$$

On the other hand, most KBs have annotated types for entities and each entity can have multiple types. For example, *Obama* has types such as *President*, *Politician*, *Attorney*, *American Citizen* etc. Taking full advantage of the entity-type information, we define T as the vocabulary of entity types, and $T(e)$ as the entity-type collections of the entity e, i.e., $T(e) = \{e^{t_i}\} | e^{t_i} \in T, i \in [1, n]$. n is the maximum number of entity types for entity e and we set $n = 8$. Given the entity-type set of the entity e, we calculate the entity-type embedding of the entity e as the average of its entity types

$$\mathbf{e}^t = \frac{1}{|T(e)|} \sum_{e^{t_i} \in T(e)} \mathbf{e}^{t_i}.$$

Then we define the transformed entity-type embeddings as $g(\mathbf{e}^t) = \mathbf{M}\mathbf{e}^t$ or $g(\mathbf{e}^t) = f(\mathbf{M}\mathbf{e}^t + \mathbf{b})$, where $\mathbf{M} \in R^{d \times k}, \mathbf{b} \in R^d$, f can be sigmoid, tanh, d and k is respectively the dimension of entity(relation) and entity type.

Having established the context and types information of the entities, we identify seven types of input path for an entity pair (e_h, e_t). The first type only involves the relations in the path, thus it is defined as $\pi = [\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_s]$. The second type considers the intermediate entities and relations along the path, which is given by $\pi = [\mathbf{e}_h, \mathbf{r}_1, \mathbf{e}_2, \ldots, \mathbf{r}_s, \mathbf{e}_t]$. The third and forth types separately obtain the entity representation by the contextual information and entity type, and are defined as $\pi = [\overline{\mathbf{e}}_h, \mathbf{r}_1, \overline{\mathbf{e}}_2, \ldots, \mathbf{r}_s, \overline{\mathbf{e}}_t]$ and $\pi = [g(\mathbf{e}_h^t), \mathbf{r}_1, g(\mathbf{e}_2^t), \ldots, \mathbf{r}_s, g(\mathbf{e}_t^t)]$ The fifth type takes the contextual information of entity, entities and relations into account, and is defined as $\pi = [\overline{\mathbf{e}}_h, \mathbf{e}_h, \mathbf{r}_1, \overline{\mathbf{e}}_2, \mathbf{e}_2, \ldots, \mathbf{r}_s, \overline{\mathbf{e}}_t, \mathbf{e}_t]$. The sixth type leverages the information from intermediate entity types, entities and relations along the path, and is defined as $\pi = [g(\mathbf{e}_h^t), \mathbf{e}_h, \mathbf{r}_1, g(\mathbf{e}_2^t), \mathbf{e}_2, \ldots, \mathbf{r}_s, g(\mathbf{e}_t^t), \mathbf{e}_t]$. The seventh type jointly considers relations, entities, entity-types and contextual information of entity and is defined as $\pi = [g(\mathbf{e}_h^t), \overline{\mathbf{e}}_h, \mathbf{e}_h, \mathbf{r}_1, g(\mathbf{e}_2^t), \overline{\mathbf{e}}_2, \mathbf{e}_2, \ldots, \mathbf{r}_s, g(\mathbf{e}_t^t), \overline{\mathbf{e}}_t, \mathbf{e}_t]$.

For all above types of input path, we define their length as the number of relations contained in the path. i.e., len(π) = s. Seven types respectively are fed into C-RNN architecture, and the respective model are denoted as C-RNN, C-RNN+E, C-RNN+C, C-RNN+T, C-RNN+E+C, C-RNN+E+T C-RNN +E+C+T. "E", "C" and "T" respectively denote "Entity", "Context" and "Types". For convenient notation, we set the input path as $\{x_1, x_2, \ldots, x_l\}$ (l is the number of entries in the path), and the vector embedding of its ith element is $\mathbf{x}_i \in R^d$. We denote $\mathbf{x} \in R^{l \times d}$ as the input path embedding for our C-RNN architecture.

4 Model

This section provides two convolutional-based recurrent neural networks for complex reasoning over knowledge bases. The first one as described in Section 4.1 extracts the local features of input path through one-layer convolutional neural network, and then feeds such features into a unidirectional recurrent neural network for learning the path representation. Different from the unidirectional C-RNN, the second one illustrated in Section 4.2 utilizes a bidirectional recurrent neural network to represent the embedding of input path.

4.1 Unidirectional C-RNN

This subsection proposes a convolutional-based unidirectional recurrent neural network named C-UiRNN for complex reasoning over objects (relations and entities) of KGs. As Fig. 3 shows, C-UiRNN is the combination of convolutional neural network and unidirectional recurrent neural network. The left box represents one-layer convolutional neural network and the right box represents unidirectional recurrent neural network.

4.1.1 *CNN for Extracting Local Features*

As the left box of Fig. 3 shows, the one-layer CNN includes a convolutional layer and local max-pooling layer. The convolutional layer extracts local correlations at different positions of a path by sliding over input path embedding using multiple filters. The local max-pooling layer obtains the valuable higher-order representations from the local correlations.

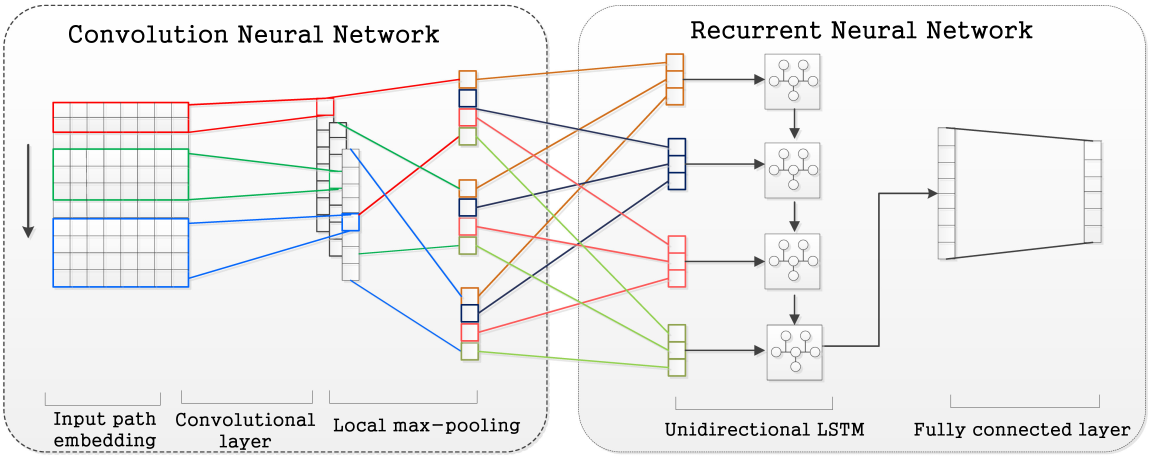

Fig. 3. Simple visualization of unidirectional C-RNN architecture.

Convolutional Layer. First at each position i of the path, we define a window $\mathbf{W}_i$ of h vectors in the path as $\mathbf{W}_i = [\mathbf{x}_i, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_{i+h-1}]$, where the commas represent the concatenation of successive row vectors $\mathbf{x}_i, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_{i+h-1}$. Then a filter $\mathbf{C} \in R^{h\times d}$ with size h for convolution operation is applied to ith window and produces a new feature m_i, which is given by

$$m_i = f(\mathbf{W}_i \cdot \mathbf{C} + b),$$

where $\cdot$ is the element-wise multiplication, $b \in R$ is a bias term and f is a nonlinear transformation function which can be sigmoid, hyperbolic tangent, etc. Therefore a feature map $\mathbf{m}$ is generated by convoluting with each position of the path, i.e.,

$$\mathbf{m} = [m_1, m_2, \ldots, m_l].$$

Here we apply paddings to the input embedding $\mathbf{x}$ so that all feature maps $\mathbf{m}$ have the same number of entries, meanwhile we utilize n filters to generate n feature maps as

$$\mathbf{M} = \{\mathbf{m}^1, \mathbf{m}^2, \ldots, \mathbf{m}^n\}.$$

Pooling Layer. We first devise a local max-pooling window with the size s, then we apply it with stride s on each feature map $\mathbf{m}^i$ and generate the pooling feature representations. For example, applying 2-size with 2-strides max-pooling to $[2, 3, 1, 2, 4, 2]$ yields $[3, 2, 4]$. Thus for n feature maps in convolutional layer, we can get n respective pooling feature maps as $\mathbf{p}^1, \mathbf{p}^2, \ldots, \mathbf{p}^n$. Further we reorganize such pooling feature maps as

$$\mathbf{P} = [\mathbf{p}^1; \mathbf{p}^2; \ldots; \mathbf{p}^n].$$

Here semicolons represent column vector concatenation and $\mathbf{p}^i \in R^{\lceil l/s \rceil}$.[2] Each row $\mathbf{P}_j$ of $\mathbf{P} \in R^{\lceil l/s \rceil \times n}$ produced from n filters is regarded as the new feature representation. Then such new successive higher-order window representations $\mathbf{P}_j (j \in [1, \lceil l/s \rceil])$ are fed into RNN, which preserve the original sequence properties of a path.

2. $\lceil x \rceil$ returns the ceiling of x as a float, i.e., the smallest integer value greater than or equal to x.

4.1.2 Unidirectional RNN for Capturing Global Features

The second stage of C-UiRNN takes as input the sequence of higher-order representations, which are the output of convolutional neural network. Unidirectional recurrent neural network utilizes Long Short-Term Memory(LSTM) [29], [30] for relation prediction, thus it is better at finding and exploiting long range dependencies in the path sequence. The following gives more details.

Unidirectional RNN. As the right box of Fig. 3 illustrates, in k-step cell of RNN, both the kth higher-order vector and the output of $k-1$-step cell of RNN are taken as inputs, and then it outputs an intermediate representation $\mathbf{h}_k$ as

$$\mathbf{h}_k = g(\mathbf{W}_{\mathbf{hh}}\mathbf{h}_{k-1} + \mathbf{W}_{\mathbf{ih}}\mathbf{P}_k),$$

where $\mathbf{W}_{\mathbf{hh}} \in R^{d\times d}$, $\mathbf{W}_{\mathbf{ih}} \in R^{d\times n}$ are the parameters of RNN. $\mathbf{h}_k$ is the input to the next step and the last output vector $\tilde{\mathbf{h}}$, i.e., $\mathbf{h}_{\lceil l/s \rceil}$ is regarded as the representation of the input path.

4.2 Bidirectional C-RNN

Besides the unidirectional convolutional-based RNN, we devise the bidirectional convolutional-based recurrent neural network, named C-BiRNN. C-BiRNN has the aforementioned convolutional neural network part and applies a bidirectional LSTM network to make full use of past feature (via forward states) and future features (via backward states).

Bidirectional RNN. As Fig. 4 illustrates, the bidirectional RNN involves forward RNN and backward RNN, which have respective weight parameters and intermediate states. Forward RNN and backward RNN are two unidirectional RNN that have input path with different directions. Therefore in k-step cell of RNN, there are two intermediate representation $\mathbf{h}_k^f$ and $\mathbf{h}_k^w$ as following:

$$\begin{aligned}
\mathbf{h}_k^f &= g(\mathbf{W}_{\mathbf{hh}}^f\mathbf{h}_{k-1}^f + \mathbf{W}_{\mathbf{ih}}^f\mathbf{P}_k) \\
\mathbf{h}_k^w &= g(\mathbf{W}_{\mathbf{hh}}^w\mathbf{h}_{k-1}^w + \mathbf{W}_{\mathbf{ih}}^w\mathbf{P}_{\lceil l/s \rceil - k+1}),
\end{aligned}$$

where the variables with the superscript f and w are respectively the parameters of forward and backward RNN, and $\mathbf{W}_{\mathbf{hh}}^f, \mathbf{W}_{\mathbf{hh}}^w \in R^{d\times d}, \mathbf{W}_{\mathbf{ih}}^f, \mathbf{W}_{\mathbf{ih}}^w \in R^{d\times n}$. The last vector representation $\mathbf{h}_{\lceil l/s \rceil}^f$ and $\mathbf{h}_{\lceil l/s \rceil}^w$ are separately the forward and backward representation of the input path. We obtain the

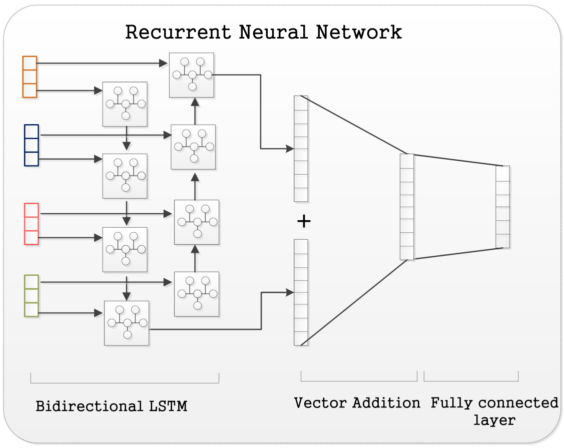

Fig. 4. Simple visualization of bidirectional C-RNN architecture.

path representation vector $\tilde{\mathbf{h}}$ by conducting add-vector operation on $\mathbf{h}^{f}_{\lceil l/s\rceil}$ and $\mathbf{h}^{w}_{\lceil l/s\rceil}$, which is given by

$$\tilde{\mathbf{h}} = \mathbf{h}^{f}_{\lceil l/s\rceil} + \mathbf{h}^{w}_{\lceil l/s\rceil}.$$

For both C-UiRNN and C-BiRNN models, finally a fully connected layer is added on the top for predicting target relation, given by

$$\tilde{\mathbf{y}} = g(\mathbf{U}\tilde{\mathbf{h}} + \mathbf{b}),$$

where we adopt $g = sigmoid$, and $\mathbf{U}$ is the weight parameter. Generally C-UiRNN and C-BiRNN take full advantages of CNN for extracting the local features of sequences and RNN for capturing global information of any length of sequences.

5 Model Training

5.1 Training

For both unidirectional and bidirectional C-RNN, we respectively train two lines of our convolutional-based recurrent neural network. The first line trains a separate C-RNN model for each relation type, denoted as C-RNN(separated), the second line shares relation representations and the relation-specific parameters of C-RNN across all target relations, denoted as C-RNN(shared). C-RNN(shared) is the multi-task learning among prediction of all target relations. Both C-RNN(separated) and C-RNN(shared) regard the output vector at the last step of C-RNN as the representation of a path, and finally connect a fully connected layer on top.

We use existing observed facts in the KB as positive samples and unobserved facts as negative samples [31], [32] to train our C-RNN(separated) and C-RNN(shared) models. For C-RNN(shared), we denote the query relation set as $\mathbf{R} = \{r_1, r_2, \ldots, r_o\}$($o$ is the number of target relations), the set of positive and negative samples as $\Delta^{+}_{\mathbf{R}}$ and $\Delta^{-}_{\mathbf{R}}$. For each positive input sample $\pi^{+} = \{x_1, x_2, \ldots, x_l\}$, we randomly replace o_1 items in π^{+} with items that are selected from the entity set or relation set to construct a negative sample π^{-}, where $o_1 = \lceil (1-\epsilon)l \rceil$, $\lceil\cdot\rceil$ is the ceiling function and $\epsilon \in [0, 1)$. Here we define the category label of a positive sample as $y^{+} \in [1, 2, \ldots, o]$, and that of a negative sample as $y^{-} \in [1, 2, \ldots, o]$. After the fully connected layer of C-RNN (shared) model, each input sample has the estimated probabilities $\tilde{\mathbf{y}}_i \in [0, 1]$ for each label $i \in \{1, 2, \ldots, o\}$. We minimize the following negative log-likelihood to train our model

$$L(\Theta_{\mathbf{R}}, \Delta^{+}_{\mathbf{R}}, \Delta^{-}_{\mathbf{R}}) = -\Big\{ \sum_{\pi^{+}\in\Delta^{+}_{\mathbf{R}}} log(\tilde{\mathbf{y}}_{y^{+}}) + \sum_{\pi^{-}\in\Delta^{-}_{\mathbf{R}}} log(1-\tilde{\mathbf{y}}_{y^{-}}) \Big\}, \quad (1)$$

where $\tilde{\mathbf{y}}_{y^{+}}$ and $\tilde{\mathbf{y}}_{y^{-}}$ are the y^{+}th and y^{-}th element of $\tilde{\mathbf{y}}$, Θ is the set of all parameters of our C-RNN(shared) model. The C-RNN(shared) neural network and classifier are optimized using RMSPROP,[3] which is an adaptive learning rate method proposed by Geoff Hinton.

For C-RNN(separated), we apply the above loss function for predicting each relation. It is a binary classification problem. Correspondingly the query relation set is defined as $\mathbf{R} = \{r_i\}(i \in [1, o])$, the set of positive and negative samples as $\Delta^{+}_{r_i}$ and $\Delta^{-}_{r_i}$, the category label as $y^{+} = 1, y^{-} = 0$ and the set of parameters as $\Theta_{\mathbf{r}_i}$.

5.2 Padding and Regularization

As fixed-length input is required in the convolution layer of our C-RNN models, we pad all paths whose length are less than *maxlen* with special symbols at the end of original path, and filter out those paths which are longer than *maxlen*. We define *maxlen* as the maximum length of path among training, valid and testing sets. For each path π with length $l(l < maxlen)$, we will pad the path π with $maxlen - l$ special embeddings at the end of original path. For example, the first type of path π with length l is set as $\{r_1, r_2, \ldots, r_l\}$, its embedding is $[\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_l]$. Then $maxlen - l$ special embeddings are indicated by the *padding* relation u_r, then the padded path is $[\mathbf{r}_1, , \mathbf{r}_2, \ldots, \mathbf{r}_l, \mathbf{u}_r, \ldots., \mathbf{u}_r]$.

For regularization, two metrics are employed in our C-RNN architecture: dropout and L2 weight regularization. We only apply dropout to the output of CNN before feeding into the RNN for preventing co-adaptation, and L2 regularization to the weight parameters of the fully connected layer. We also initialize the embeddings of entities, entity types and relations from the uniform distribution [-0.25, 0.25].

6 Experiment

There are two lines of KG complex reasoning methods, including separated models and shared models. The separated models train a separate model for predicting each target relation. The shared models train a shared model that shares the relation embeddings and parameters of the relation-specific separated models. Therefore we evaluate our C-RNN(separated) and C-RNN(shared) models on typical predictive paths task with FreeBase+ClueWeb dataset, and compare them with several baseline models. Experimental results show that our models achieve consistent improvements on predictive paths task.

DataSets. FreeBase+ClueWeb[4] is released in [19], which is a subset of FreeBase [1] enriched with information from ClueWeb [33]. The dataset embodies 46 relation types in

3. http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf
4. http://iesl.cs.umass.edu/downloads/inferencerules/release.tar.gz

TABLE 1
Statistics of the Dataset

Statistics	#
# FreeBase relation types	27,791
# textual relation types	23,599
# query relation types	46
# entity pairs	3.22M
# unique entity types	2218
Avg. path length	4.7
Max path Length	7
Total # paths	191M

FreeBase that have the most number of instances. For each relation type r, a set of triplet facts (e_h, r, e_t) and a set of paths between the entity pair (e_h, e_t) in the KB are extracted from ClueWeb sentences that contain two entities linked to FreeBase. The relation type is formed from the raw text between the two entities in the sentences. To limit the length of relation types, only two following words after the first entity and two words before the second entity are retained in FreeBase+ClueWeb. The entity type information is also collected from FreeBase. Table 1 demonstrates the important statistics of the dataset.

Baselines. Most KG reasoning methods are based on either path formulas or KG embedding. We explore methods from both of these two classes in our experiments. For embedding-based models, we mainly compare with several typical and promising methods for knowledge graph completion, including TransE [9], TransH [10], TransR [34], TransD [11], RESCAL [13], DistMult [35] and ComplEx [36]. Such models only learn from direct relations between entities but ignore multiple-step relation paths, which also contain rich inference patterns between entities. For those models, we first reorganize the path $\{(e_h, r_1, e_2, r_2, \ldots, e_s, r_s, e_t)\}$ involved in data set into some fact triplets as $\{(e_h, r_1, e_2), (e_2, r_2, e_3), \ldots, (e_s, r_s, e_t)\}$, and then we train two lines of entity and relation embeddings. The first line learns separate entity and relation embeddings for each reasoning task, the second learns shared embeddings for all relation reasoning tasks. We rerun those models using the public available code[5] and each model is trained for 500 epochs. For path-based methods, aforementioned methods in "Related work" including PRA, PRA Classifier, Cluster PRA Classifier, Composition-Add, Path-RNN, PRA Classifier-b, Cluster PRA Classifier-b, Path-RNN+PRA Classifier, Path-RNN+PRA Classifier-b and Single-RNN are compared with our C-RNN methods.

Parameters Setup. We implement our models based on Tensorflow, which supports efficient symbolic differentiation and transparent use of a GPU. To benefit from the efficiency of parallel computation of the tensors, we train the model on a GPU. We adopt the same dataset used in [19], [20] to evaluate our C-RNN models. There are four convolutional-based recurrent neural networks for training, including C-UiRNN (separated), C-UiRNN(shared), C-BiRNN(separated) and C-BiRNN(shared). The word "separated" identifies such model trains a separated model for each target relation. The word "shared" means such model learns shared embeddings and parameters across all target relations.

5. https://github.com/thunlp/TensorFlow-TransX and https://github.com/mana-ysh/knowledge-graph-embeddings

TABLE 2
Parameter Configuration on Convolutional Filter Size

Input Path	Relation	+E/ +C/ +T	+E+C/ +E+T	+E+C+T
Filter Size	{2,3,4}	{3,4,5}	{5,6,8}	{7,8,11}

For our four C-RNN models, we select the dimension of entity(relation) embeddings and RNN hidden states d from $\{50, 100, 200, 300\}$, the dimension of entity type k from $\{50, 100, 200, 300\}$, the size of convolutional filter h from $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$, the number of filters for each h is set to 32, the size and stride of local max-pooling s from $\{1, 2, 4\}$, and the probability of dropout p from $\{0.2, 0.5, 0.8\}$, the weight of L2 regularization w from $\{0.001, 0.01, 0.2, 0.5, 0.8\}$, the learning rate α from $\{0.0001, 0.001, 0.01, 0.1\}$, the batch size b from $\{128, 256, 512, 1200, 2400, 4800\}$. The optimal parameters that make our models achieve the highest reasoning performance are determined by validation set, i.e., in training stage, we use the validation set to evaluate our C-RNN model every 100 epoch and preserve the optimal parameters that achieves the best performance on validation set. Each epoch of our model trains a batch of the training set.

Metric. We use the average precision(AP) for our evaluation metric. The MAP is the mean AP across all query types, which approximates the area under the Precision Recall curve [37].

6.1 Results of Separated Models

In this experiment, to compare with many separated models, we respectively run our separated unidirectional C-RNN and bidirectional C-RNN with seven types of input path referred in Section 3. The results of previous baselines are referred from their report since the same data set is used.

Parameters. For unidirectional C-RNN models, the optimal parameter setting is $\alpha = 0.001$, $d = 300$, $k = 300$, $s = 4$, $p = 0.5$, $w = 0.001$, $b = 4800$. For bidirectional C-RNN models, the optimal parameter setting is $\alpha = 0.001$, $d = 200$, $k = 200$, $s = 4$, $p = 0.5$, $w = 0.01$, $b = 256$. For both unidirectional and bidirectional C-RNN models with seven types of input path, the configuration of convolutional filter size h are shown in Table 2.

Analysis. Table 3 gives the experimental results, which presents the comparison of our C-BiRNN, C-BiRNN +E+T+C and previous baselines in the top part, and the comparison of our variants and state-of-the-art Path-RNN. The MAP in Table 3 is the average value of MAP of 46 query relations since each query relation has a MAP value. The column "Increase" in Table 3 is the increase of MAP comparing our best model with the baselines in the first column. From Table 3 we can see that: (1) Our C-UiRNN and C-BiRNN separated models outperform state-of-the-art separated models Path-RNN and PRT+T. It suggests that the combination of CNN and RNN greatly improves performance on knowledge graph reasoning. (2) In our separated C-RNN models, the bidirectional models C-BiRNN perform better than the unidirectional models C-UiRNN since C-BiRNN uses not only the past but also the future input features for learning the path representation.

TABLE 3
Model Performance(SEPARATED)

Model	MAP	Increase
TransE [9]	21.0	+64.6
TransH [10]	27.3	+58.3
TransR [34]	32.2	+53.4
TransD [11]	39.4	+46.2
RESCAL [13]	15.7	+69.9
DistMult [38]	45.9	+39.7
ComplEx [36]	50.5	+35.1
PRA [18]	49.6	+36.0
PRA Classifier [39]	51.3	+34.3
Cluster PRA Classifier [40]	53.2	+32.4
Composition-Add [41]	45.4	+40.2
Path-RNN-*random* [19]	56.9	+28.7
Path-RNN [19]	57.0	+28.6
PRA Classifier-b [19]	58.1	+27.5
Cluster PRA Classifier-b [19]	58.0	+27.6
Path-RNN+PRA Classifier [19]	58.4	+27.2
Path-RNN+PRA Classifier-b [19]	61.2	+24.4
ALL-PATH+NODES [42]	62.6	+23.0
PRA + T [19]	64.2	+21.4
C-BiRNN(separated)	**68.5**	+17.1
C-BiRNN+E+T+C(separated)	**85.6**	-
Variants of C-RNN(separated)	**MAP**	**Increase**
Path-RNN [19]	57.0	+28.6
PRA + T [19]	64.2	+21.4
C-UiRNN(separated)	66.9	+18.7
C-BiRNN(separated)	68.5	+17.1
C-UiRNN+E(separated)	80.8	+4.8
C-BiRNN+E(separated)	82.5	+3.1
C-UiRNN+C(separated)	80.3	+5.3
C-BiRNN+C(separated)	82.0	+3.6
C-UiRNN+T(separated)	79.4	+6.2
C-BiRNN+T(separated)	80.9	+4.7
C-UiRNN+E+C(separated)	81.8	+3.8
C-BiRNN+E+C(separated)	82.5	+3.1
C-UiRNN+E+T(separated)	82.3	+3.3
C-BiRNN+E+T(separated)	83.4	+2.2
C-UiRNN+E+T+C(separated)	**84.1**	+1.5
C-BiRNN+E+T+C(separated)	**85.6**	-

Detailedly seven separated C-BiRNN models have 1.59, 1.74, 1.69, 1.46, 0.72, 1.07 and 1.52 percent higher than respective separated C-UiRNN models. (3) Models that consider entities, the contextual information or entity types are more competitive than the shallow models that only consider the relations in the path. It indicates that entities, entity types and the contextual information can provide indispensable and effective characteristic information for learning path representation. (4) The path-based KG reasoning methods such as C-BiRNN are more promising and effective than embedding-based models such as TransE, ComplEx. The reason is that embedding-based models do not use the multi-hop path information to perform KG reasoning. Even path-based model PRA with the lowest performance(49.6 percent) can be comparable with the embedding-based model ComplEx(50.5 percent) that has higher performance.

6.2 Results of Shared Models

In this experiment, we run our shared C-RNN models with seven types of input path, and compare them with state-of-the-art shared model Single-RNN [20] and several embedding-based models, such as TransE [9], TransH [10], TransR [34], TransD [11], RESCAL [13], DistMult [35] and ComplEx [36].

TABLE 4
Model Performance(SHARED)

Model	MAP	Increase
TransE [9]	27.9	+52.5
TransH [10]	30.0	+ 50.4
TransR [34]	37.3	+ 43.1
TransD [11]	43.1	+37.3
RESCAL [13]	20.7	+59.7
DistMult [35]	54.5	+25.9
ComplEx [36]	65.4	+10.3
Single-RNN [20]	70.11	+9.0
Single-RNN+E [20]	71.4	+7.2
Single-RNN+T [20]	73.2	+8.2
Single-RNN+E+T [20]	72.2	+9.2
C-BiRNN(shared)	71.2	-
C-BiRNN+E+C+T(shared)	**80.4**	
Variants of C-RNN(shared)	**MAP**	**Increase**
Single-RNN+T [20]	73.2	+7.2
Single-RNN+E+T [20]	72.2	+8.2
C-UiRNN(shared)	70.8	+9.6
C-BiRNN(shared)	71.2	+9.2
C-UiRNN+E(shared)	72.0	+8.4
C-BiRNN+E(shared)	73.8	+6.6
C-UiRNN+C(shared)	74.1	+6.3
C-BiRNN+C(shared)	75.1	+5.3
C-UiRNN+T(shared)	74.7	+5.7
C-BiRNN+T(shared)	76.8	+3.6
C-UiRNN+E+C(shared)	76.2	+4.2
C-BiRNN+E+C(shared)	77.5	+2.9
C-UiRNN+E+T(shared)	77.2	+3.2
C-BiRNN+E+T(shared)	78.1	+2.3
C-UiRNN+E+C+T(shared)	79.7	+0.7
C-BiRNN+E+C+T(shared)	**80.4**	-

Parameters. For unidirectional C-RNN models, the optimal parameter setting is $\alpha = 0.001$, $d = 300$, $k = 300$, $s = 2$, $p = 0.2$, $w = 0.5$, $b = 2400$. For bidirectional C-RNN models, the optimal parameter setting is $\alpha = 0.001$, $d = 300$, $k = 300$, $s = 2$, $p = 0.5$, $w = 0.001$, $b = 128$. For both C-RNN models, the configuration of convolutional filter size is illustrated in Table 2.

Analysis. The results of our shared models are shown in Table 4. Similar to Table 3, we also compare our C-BiRNN, C-BiRNN+E+T+C with baselines on the top part of Table 4, and compare the variants of our C-RNN with state-of-the-art shared model Single-Path on the bottom part of Table 4. From this table, we can observe that: (1) Both our shared C-UiRNN and C-BiRNN models significantly outperform all baselines. It indicates that cooperating the convolution neural network with the recurrent neural network is beneficial for representing paths across all target relations. (2) Incorporating entities and its additional information into reasoning methods are beneficial for complex reasoning on knowledge bases, since entities and its additional information provide rich sematic and logic information. Specifically, C-UiRNN (shared) with +E, +C or +T is at least 1.1 percent higher than C-UiRNN(shared), and C-BiRNN(shared) with +E, +C or +T has at least 1.9 percent higher than C-BiRNN. (3) Each C-UiRNN(shared) has worse performance than its respective

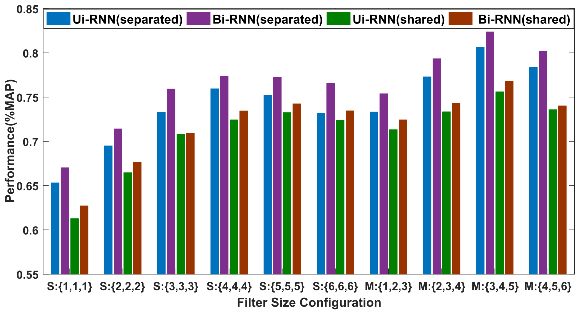

Fig. 5. Performance on different filter size configuration.

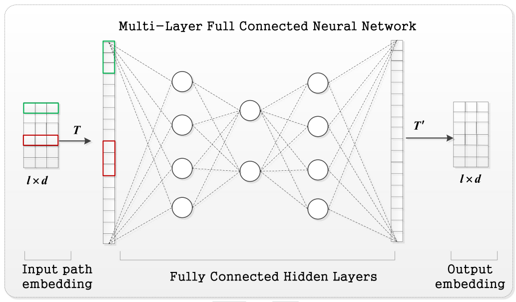

Fig. 6. Simple visualization of N-RNN framework.

C-BiRNN(shared), which implies that the bidirectional RNN is helpful to improve the reasoning performance. (4) Even C-UiRNN(shared) and C-BiRNN(shared) are comparable with SingleRNN+E, SingleRNN+T and SingleRNN+E +T. This indicates the shared C-UiRNN and C-BiRNN models are more feasible and capable for learning the representations of paths. (5) For shared reasoning models, path-based reasoning methods have more desirable results than embedding-based methods. The lower performance of embedding-based models is caused by failing to utilize the multi-hop path for KG reasoning. The highest performance of embedding-based method is just 65.4 percent, which is at least 4.8 percent lower than those path-based reasoning methods.

In general, our separated and shared C-RNN models achieve at least 9 percent higher performance on MAP than other models. The three advantages of our model give its superiority of reasoning performance: (1) We consider the local correlation information and the sequential information of the paths for complex reasoning task. (2) We use the convolution neural network for extracting the local features and recurrent neural network for capturing the sequential features. (3) We incorporate the entities, entity-types and context knowledge to provide richer semantic characteristics of paths for reasoning.

7 Discussion

7.1 Performance in Using of CNN

This section discusses how the using of CNN for RNN is effective for capturing the local information of input paths.

In our C-RNN models, the filters of convolutional neural network are devised to capture the local n-gram information of input paths. Therefore we investigate the impact of different filter configurations in the convolutional layer on the model performance. There are two cases: (a) single convolutional layer with the same filter size, and (b) single convolutional layer with different size of filters. We investigate filter size h from $\{1,2,3,4,5,6\}$ in above cases, and set the number of filters to 32, the filter length to 3. Each n-gram window is transformed into $3*32$ convoluted features after convolution, and the sequence of window representations after pooling operation is fed into RNN. We show in Fig. 5 the prediction performance on "+Entity" setting. In Fig. 5, S means single convolutional layer with the same filter size, and M means single convolutional layer with different filter sizes. Fig. 5 finds that (1) Filter configuration $S:\{1,1,1\}$ has the worst performance than any other filter configurations due to the filter size as 1 only captures the *local* 1-gram information of the input path. (2) Single convolutional layer with different filter sizes performs better than single convolutional layer with the same size filters. That is different filter sizes could exploit features of different local n-grams information. (3) Filter size as $\{3,4,5\}$ achieves the highest performance. Since the path is Entity+Relation, then the sliding window with size $\{3,4,5\}$ can naturally capture the correlation among the sub-paths $(e_1, r_1, e_2), (e_1, r_1, e_2, r_2), (e_1, r_1, e_2, r_2, e_3)$. Note that we also observe the similar phenomenon in C-RNN model with the other types of input path.

Additionally, it is shown that the combination of CNN and RNN has better superiority than other frameworks for capturing the local information. We first devise another framework named N-RNN for experiments that replaces the CNN component of C-RNN with a multi-layer full connected neural network. As the Fig. 6 shown, for the multi-layer full connected neural network, N-RNN first takes the embedding of path as input and feeds the output of the last layer into RNN. In Fig. 6, $T:\mathbf{A}\rightarrow\mathbf{a}$ represents that it reshapes the matrix $\mathbf{A}$ into a vector $\mathbf{a}$ according to row-major order. $T^{'}:\mathbf{a}\rightarrow\mathbf{A}$ is the inverse operation of T, l is the length of input path and d is the dimension of embedding. In the experiment, we set the number of hidden layers as $L=\{1,2,3\}$, and denote the number of elements of a matrix or a vector as $ele(.)$. For $L=\{1,2,3\}$, we separately set the size of hidden layers as $\{\frac{ele(\mathbf{x})}{2}\}, \{\frac{ele(\mathbf{x})}{2}, \frac{ele(\mathbf{x})}{2}\}, \{\frac{ele(\mathbf{x})}{2}, \frac{ele(\mathbf{x})}{4}, \frac{ele(\mathbf{x})}{2}\}$. $\mathbf{x}$ is the input embedding. For all above settings, we also train the separated N-RNN and shared N-RNN by using the non-linearity activate function $sigmoid$. Experimental results illustrated in Table 5 can see that for both separated and shared models, C-UiRNN and C-BiRNN achieve consistently higher performance than N-RNN with 1, 2, 3 layer. It indicates that the combination of CNN and RNN is more beneficial to capture local correlation of input paths than fully connected neural network. Moreover, for a more detailed and intuitive comparison of the experimental results on these models, we separately give the performance of 46 relations on separated and shared models under "+Entity" setting in Fig. 7, which suggests that C-RNN obtains better performance than N-RNN on each relation.

7.2 Performance in Different Length of Path

This section explores the effect of different length of path to the performance in representing path embedding. In this experiment, we compare the performance of our C-RNN

TABLE 5
Performance in Using of Convolutional Neural Network

Model		Relation	+E	+C	+T	+E+C	+E+T	+E+C+T
Separated Model	N-RNN(1-layer)	65.0	72.8	74.6	74.1	75.9	76.7	77.4
	N-RNN(2-layer)	62.8	70.1	71.2	73.7	73.4	75.8	76.9
	N-RNN(3-layer)	61.5	69.5	70.4	71.5	70.2	72.2	74.3
	C-UiRNN	**66.9**	**80.8**	**80.3**	**79.4**	**81.8**	**82.3**	**84.1**
	C-BiRNN	**68.5**	**82.5**	**82.0**	**80.9**	**82.5**	**83.4**	**85.6**
Shared Model	N-RNN(1-layer)	61.8	65.4	66.7	68.9	69.5	69.1	69.8
	N-RNN(2-layer)	60.3	63.2	65.1	67.5	69.0	68.9	68.7
	N-RNN(3-layer)	60.4	62.1	63.3	65.1	67.2	66.34	65.2
	C-UiRNN	**70.8**	**72.0**	**74.1**	**74.7**	**76.2**	**77.2**	**79.7**
	C-BiRNN	**71.2**	**73.8**	**75.1**	**76.8**	**77.5**	**78.1**	**80.4**

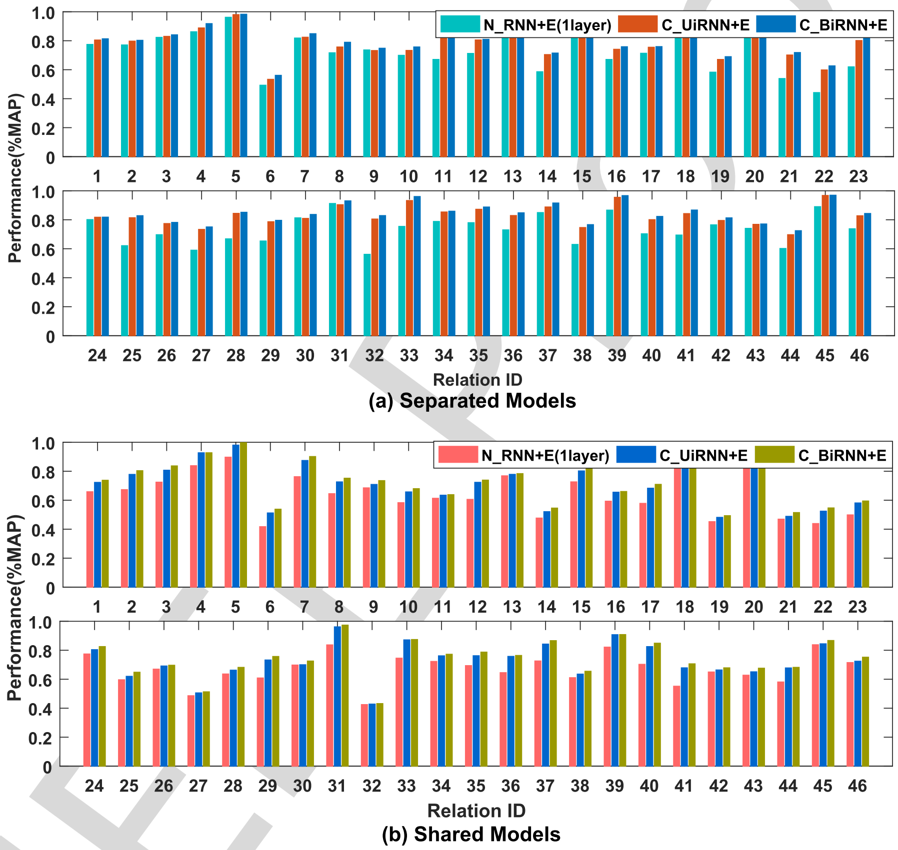

Fig. 7. Performance of 46 relations under "+Entity" setting.

models with state-of-the-art multi-hop separated and shared models, i.e., Path-RNN and SingleRNN+E+T. We generate the training, valid data according to different restrictions on length of paths. For example, we restrict the length as 3, only paths in training and valid set whose length are not more than 3 are selected, and other paths are removed. We give the restrictions of length as $\{1, 2, 3, 4, 5, 6, 7\}$ and produce seven training and valid sets. Then we only evaluate our C-UiRNN(separated), C-BiRNN(separated), C-UiRNN(shared)+E+T and C-BiRNN(shared)+E+T models and get the respective performance in test data as Fig. 8. As the Fig. 8 shows, we can see that: (1) For separated and shared reasoning models in (a) and (b), our C-UiRNN and C-BiRNN respectively outperform separated model Path-RNN and shared model Single-RNN+E+T. (2) In both (a) and (b), the bidirectional C-RNN models are also superior to the unidirectional C-RNN. (3) The performance is increasing with growth of the length, i.e., the training set with extra training examples of longer paths leads the better performance.

On the other hand, we also evaluate the number distribution of the true labeled samples on different lengths under the length restriction as 7. The results of separated and shared models are given in Fig. 9. From Fig. 9 we can see for all multi-hop reasoning models, the distribution peaks at lengths $= \{3, 4, 5\}$, i.e., the true labeled samples determined

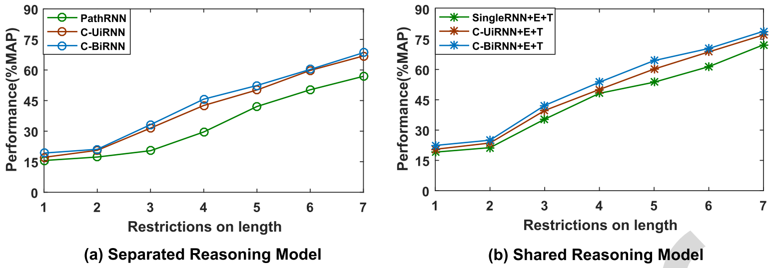

Fig. 8. Performance in training data with different length restrictions.

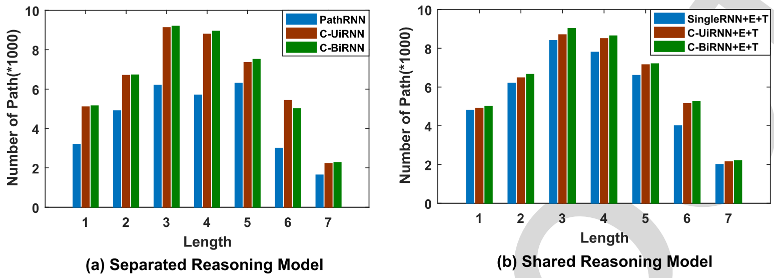

Fig. 9. Length distribution of true labeled relation prediction.

by the paths of lengths $= \{3, 4, 5\}$ have a large proportion on the overall true labeled samples. Such all above observations from Figs. 8 and 9 suggest that methods which limit the length to a small value might restrict the performance and generalization.

7.3 Performance in Limited Data Regime

The dataset FreeBase+ClueWeb contains 46 relation types that are constructed from reasonable amounts of data. Usually we have very limited data in many important applications. To simulate this common scenario, we follow [19] to construct a small dataset to evaluate our model's performance. Specifically, we first randomly select 23 out of 46 relation types. Then we remove all but 1 percent of the positive and negative triplets previously used for training.

In this experiment, we also compare our models with state-of-the-art separated model Path-RNN and shared model Single-RNN. Therefore we rerun Path-RNN and SingleRNN[6] with our produced small data, and obtain the similar results to the results published in [20]. We also run our separated and shared C-UiRNN and C-BiRNN model with the generated small dataset. Experimental results are listed in Table 6. The observations from Table 6 are: (1) All our separated and shared models achieve much higher performance than Path-RNN and Single-RNN in the small data, which proves that our C-RNN architectures have stronger adaptability for sparse data. (2) With separated and shared settings, the bidirectional C-RNN has greater superiority than unidirectional C-RNN. This is mainly because that the bidirectional C-RNN not only utilizes the future information of path but also past information of path for reasoning on KGs. (3) Effectively under this sparse and small data, the shared models perform better than the separated models since the shared models provide additional regularization with the multi-task learning. All observations indicate that our C-RNN models have the practicality and reliability for application to KBs with sparse triplet facts.

6. https://github.com/rajarshd/ChainsofReasoning

7.4 Performance in Different Initialization of Embeddings

For all above experiment of our C-RNN models, we initialize the embeddings of entities, entity types and relations from the uniform distribution since all compared baselines are initialized from random uniform distribution or normal distribution. However, this section explores the influence of input embedding with different initialization metrics on our

TABLE 6
Model Performance When Trained With a Small Fraction of the Data

Separated Model	MAP(%)
Path-RNN	21.9
C-UiRNN(separated)	**44.3**
C-BiRNN(separated)	**45.9**
Shared Model	**MAP(%)**
Single-RNN	42.86
C-UiRNN(shared)	**50.7**
C-BiRNN(shared)	**51.3**
Single-RNN+E	62.63
C-UiRNN(shared)+E	**64.5**
C-BiRNN(shared)+E	**65.4**
Single-RNN+T	64.24
C-UiRNN(shared)+T	**66.1**
C-BiRNN(shared)+T	**67.8**
Single-RNN+E+T	63.27
C-UiRNN(shared)+E+T	**68.2**
C-BiRNN(shared)+E+T	**69.1**

TABLE 7
Model Performance on Different Initialization of Embeddings

Separated Model	MAP(%)
C-UiRNN(separated)+E-*uniform*	80.8
C-BiRNN(separated)+E-*uniform*	82.5
C-UiRNN(separated)+E-*TransE*	81.3
C-BiRNN(separated)+E-*TransE*	83.1
C-UiRNN(separated)+E-*TransH*	83.4
C-BiRNN(separated)+E-*TransH*	84.1
C-UiRNN(separated)+E-*TransD*	**84.7**
C-BiRNN(separated)+E-*TransD*	**86.0**
Shared Model	**Performance**
C-UiRNN(shared)+E-*uniform*	72.0
C-BiRNN(shared)+E-*uniform*	73.8
C-UiRNN(shared)+E-*TransE*	72.7
C-BiRNN(shared)+E-*TransE*	74.2
C-UiRNN(shared)+E-*TransH*	74.4
C-BiRNN(shared)+E-*TransH*	75.6
C-UiRNN(shared)+E-*TransD*	**75.9**
C-BiRNN(shared)+E-*TransD*	**77.0**

C-RNN models. Here we select three extra modes including *TransE*, *TransH* and *TransD* for initialization. Specifically *TransE*, *TransH* and *TransD* mode respectively indicate that the input embeddings are initialized with the embeddings pre-trained by TransE, TransH and TransD models. Among the three selected models, according to the definition of their score function referred in Section 8, TransE is very effective for 1-to-1 relation, but it has problems in complex relations such as n-to-1, 1-to-n and n-to-n relations. Although TransH introduces relation hyperplanes to extend modeling flexibility, but similar to TransE it still embeds entities and relations into the same vector space. TransD embeds entities and relations into distinct semantic spaces and simultaneously considers the multiple types of entities and relations. Intuitively, our C-RNN with *TransD* mode should have better performance than other two modes. Therefore for our C-RNN architecture we just select the separated and shared C-RNN + E for evaluation. Table 7 gives the results. Both separated and shared C-RNN + E models with *TransD* mode achieve the best performance. Compared with it, the C-RNN models under *TransE* and *TransH* modes has slightly worse performance.

Based on all the above experiments, we make following conclusions: (1) For both separated and shared C-RNN models, the bidirectional C-RNN performs better than unidirectional C-RNN. (2) Although the separated C-RNN models have greater performance than shared C-RNN models, the shared C-RNN models have fewer parameters and better practicability for populating large-scale knowledge graphs. (3) The performance and practicability of our C-RNN models might be restricted by limiting the length of path to a small value. (4) Under sparse and small data, our C-RNN models still have reliability for application to KBs. (5) The performance of the model on complex reasoning can benefit from different initialization of relation and entity embeddings.

TABLE 8
Time Complexity

Model	Time Complexity
PRA	$O(N_r(N_e d + N_r d + \theta_1))$
C-RNN(separated)	$O(N_r(N_e d + N_r d + \theta_2))$
C-RNN(shared)	$O(N_e d + N_r d + \theta_3)$

7.5 Time Complexity

The separated reasoning models like PRA train a separate model for each relation type, and have the respective parameters for each relation type. The shared reasoning models, such as our C-RNN, train a shared model that shares the parameters across all relation types. For the reasoning models, all embeddings of entities $\{\mathbf{e}_i\}_{i=1}^{N_e}$, relations $\{\mathbf{r}_k\}_{k=1}^{N_r}$ and relation-specific parameters θ need to be learned. Thus the time complexity of the separated and shared reasoning models are respectively $O(N_r(N_e d + N_r d + \theta))$ and $O(N_e d + N_r d + \theta)$. We make the comparison of time complexity between our model and PRA in Table 8. In this table, N_e and N_r are the number of entities and relations, d is the dimension of entity and relation embeddings. θ_1, θ_2 and θ_3 are respectively the relation-specific parameters of PRA, C-RNN(separated) and C-RNN(shared). After analysing, we have assumed that $\theta_1 \gg \theta_2 \approx \theta_3$. From Table 8, we can see that the separated models have higher time complexity with the increase of N_r than the shared models. On the other hand, the gap of time complexity between PRA and C-RNN(separated) also increases with N_r increasing because of $\theta_1 \gg \theta_2$.

8 Related Work

8.1 Knowledge Graph Embeddings

As an emerging research direction, knowledge graph embedding for modeling multi-relation data from knowledge bases develops quickly in recent years [9], [10], [11], [12], [13], [43], [44]. From a representation learning perspective, all these methods attempt to model entities and relations of a knowledge graph into continuous vector spaces while preserving KGs inherent properties. Usually entities are represented as vectors [9], [10], [11], [12], [45] or multivariate Gaussian distribution [46], relations as vectors [9], [10], [11], matrices [47], tensors [12] or mixture of Gaussian distribution [44], [46]. For each triplet in KGs, all these methods customarily devise various scoring functions defined by vector or matrix operation to measure the plausibility of the triplet in that space. These models can be roughly classified into two groups: (1) Translational Distance-Based (TD) Models [9], [10], [11], [43], [44], [48], [49], [50], [51], (2) Semantic Matching-Based (SM) Models [12], [13], [16], [36], [38], [52], [53], [54]. The former utilizes distance between the latent representations of entities and relations to define score function, the latter utilizes similarity ones. All models usually learn entity and relation embeddings by minimizing the total plausibility of all observed triplets in a knowledge graph. Typically a margin-based ranking loss is used to conduct the minimization optimization. Recently an upper-limited scoring loss [55] is proposed to address the issue of margin-based ranking loss that can not ensure enough lower scores for observed triples.

Among TD-based models, TransE [9] regards relation embedding $\mathbf{r}$ as a translation operation from head entity embedding $\mathbf{h}$ to tail entity embedding $\mathbf{t}$, its score function is $f(h,r,t) = \|\mathbf{h} + \mathbf{r} - \mathbf{t}\|_2^2$. TransH [10] projects entity embeddings into relation-determined hyper-planes to enable an entity to have distinct representations when involved in different relations. It models each relation r as a translation vector $\mathbf{r}$ between the projected head and tail entity embeddings, and the score function is $f(h,r,t) = \|\mathbf{h}_\perp + \mathbf{r} - \mathbf{t}_\perp\|_2^2$, $\mathbf{h}_\perp = \mathbf{h} - \mathbf{w}_r^T\mathbf{h}\mathbf{w}_r$, $\mathbf{t}_\perp = \mathbf{t} - \mathbf{w}_r^T\mathbf{t}\mathbf{w}_r$, $\|\mathbf{w}_r\| \in R^d$ is the normal vector of the hyperplane. TransR [34] projects the entity embeddings $\mathbf{h}$ and $\mathbf{t}$ from the entity space to relations space with relation-specific transfer matrix $\mathbf{M}_r \in R^{d\times k}$, then its score function satisfies $f(h,r,t) = \|\mathbf{h}\mathbf{M}_r + \mathbf{r} - \mathbf{t}\mathbf{M}_r\|_2^2$. TransD [11] is an improvement of TransR, which considers the multiple types of entities and relations simultaneously, and the score function is defined as $f(h,r,t) = \|\mathbf{h}\mathbf{M}_{rh} + \mathbf{r} - \mathbf{t}\mathbf{M}_{rt}\|_2^2$, $\mathbf{M}_{rh} = \mathbf{r}_p\mathbf{h}_p^T + \mathbf{I}^{d\times k}$, $\mathbf{M}_{rt} = \mathbf{r}_p\mathbf{t}_p^T + \mathbf{I}^{d\times k}$ are the product of two projection vectors of an entity-relation pair.

Among SM-based models, RESCAL [13] models each entity as a vector to represent its latent semantics, and each relation r as a matrix $\mathbf{M}_r \in R^{d\times d}$ to model pairwise interactions between latent semantics of entities. For each triplet (h,r,t), the score function is defined as $f(h,r,t) = \mathbf{h}^T\mathbf{M}_r\mathbf{t} = \sum_{i=0}^{k-1}\sum_{j=0}^{k-1}[\mathbf{M}_r]_{ij}[\mathbf{h}]_i[\mathbf{t}]_j$. DistMult [35] restricts M_r as diagonal matrices to reduce the number of relation parameters. For each relation r, it introduces a vector embedding $\mathbf{r}$ and requires $\mathbf{M}_r = diag(\mathbf{r})$. Its score function is $f(h,r,t) = \mathbf{h}^T diag(\mathbf{r})\mathbf{t} = \sum_{i=0}^{d-1}[\mathbf{r}]_i \cdot [\mathbf{h}]_i \cdot [\mathbf{t}]_i$. ComplEx [36] considers the asymmetric relations and introduces complex-valued embeddings, i.e., the embeddings of entities and relations lie in complex space, hence the score function is $f(h,r,t) = Re(\mathbf{h}^T diag(\mathbf{r})\bar{\mathbf{t}}) = Re(\sum_{i=0}^{d-1}[\mathbf{r}]_i \cdot [\mathbf{h}]_i \cdot [\bar{\mathbf{t}}]_i)$, where $\bar{\mathbf{t}}$ is the conjugate of $\mathbf{t}$ and $Re(\cdot)$ means taking the real part of a complex value.

Although these methods have achieved great success in knowledge graph completion tasks, most of them fail to model multi-hop relation paths, which are indispensable for more complex reasoning tasks.

8.2 Multi-Hop Reasoning

PRA [18] utilizes random walks to extract the paths between an entity pair, and regards them as features for a per-target-relation binary classifier. PRA *Classifier* [39] is an extension of PRA, which exploits the observed text patterns to amplify the KB-scheme relations. PRA+T [19] is a method that incorporates the entity-types into PRA [18] for performing the random walks. Different from PRA *Classifier*, for the relation types from the ClueWeb triplets, Cluster PRA *Classifier* [40] replaces them with their respective cluster membership obtained by k-means clustering in KB. PRA *Classifier-b* [19] is an extension of PRA *Classifier* by additionally using bigrams in the path as features. The same strategy used in PRA *Classifier* applies to Cluster PRA *Classifier* and constructs a new model named Cluster PRA *Classifier-b* [19]. *Composition-Add* model [41] introduces a composition function defined by a simple element-wise additive interaction between relation matrices in the path. However *Composition-Add* ignores the fact that an entity pair usually contains multiple paths, and only models a single path. To address the issues, ALL-PATH+NODES [42] simultaneously both considers the intermediate entities in the path and performs relation reasoning on the multiple paths between an entity pair.

Recently Path-RNN [19] outperforms the above models. It learns the sequential information of paths with recurrent neural network(RNN), and models path representations as the last hidden state of RNN. Path-RNN + PRA Classifier [19] is the combination of predictions of RNN and PRA Classifier, which empowers the score of a triplet fact as the sum of their ascending rank in the two models. Also Path-RNN + PRA Classifier-b [19] conducts the combination of the predictions of RNN and PRA Classifier-b by using the above rule described previously. However such above models train a separate model for each relation and usually produce a large number of parameters for training. Single-RNN [20] shares the relation vector representations and parameters of relation-specific RNN to model a lesser number of parameters. Simultaneously it learns and reasons about the chains of relations, entities and entity types. Moreover Single-RNN introduces three score pooling methods as described in Section 2 to reduce the computation of parameters. To the best of our knowledge, SingleRNN achieves state-of-the-art performance in the relation extraction task.

9 Conclusion and Future Work

This paper proposes a convolutional-based recurrent neural network architecture named C-RNN, which is an unified architecture composed of convolutional neural network (CNN) and recurrent neural network (RNN). C-RNN first extracts the higher-order local correlation of a path through CNN, and then feeds such correlation into RNN for modeling the representation of a path. Our C-RNN effectively captures both the local correlation and global sequential information of the input path. We evaluate our C-RNN model on the typical dataset FreeBase+ClueWeb. Experimental results show that both C-RNN(separated) and C-RNN(shared) achieve improvement on reasoning over entities, entity types and relations in KBs.

As future work, we would like to pursue the following directions:

- Usually a large-scale knowledge base has multiple paths between an entity pair, and such paths can provide more evidence for inferring new predictions. In its current version, C-RNN just takes only a single path as evidence for a prediction. In order to address the diversity of paths between an entity pair, a possible solution is that we can introduce an *attention* mechanism on recurrent neural network to conduct complex reasoning on multiple paths between an entity pair.
- Instead of the convolutional neural network in our C-RNN models, we would use tree-structured convolutions or sequence-based convolutions for capturing richer higher-level local correlation of paths. Intuitively, LSTM with the above local correlation will be more effective to conduct reasoning over multi-step paths.
- The information used in C-RNN is entities, relations, entity types and the context of entities. We would try to learn entity embeddings with fusion of external

information from other data sources, e.g., incorporation of textual descriptions for entities, logical rules (first-order Horn clauses) or entity attributions.

ACKNOWLEDGMENTS

This work is supported by National Key R&D Program 2016 (No.2016YFB0801300), and the National Natural Science Foundation of China (No.61202226). We thank all anonymous reviewers for their constructive comments.

REFERENCES

[1] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: A collaboratively created graph database for structuring human knowledge," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2008, pp. 1247–1250.

[2] G. A. Miller, "WordNet: A lexical database for english," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[3] F. M. Suchanek, G. Kasneci, and G. Weikum, "YAGO: A core of semantic knowledge," in *Proc. 16th Int. Conf. World Wide Web*, 2007, pp. 697–706.

[4] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. H. Jr, and T. M. Mitchell, "Toward an architecture for never-ending language learning," in *Proc. 24th AAAI Conf. Artif. Intell.*, 2010, pp. 1306–1313.

[5] J. Lehmann, R. Isele, and M. Jakob, "DBpedia: A large-scale, multilingual knowledge base extracted from Wikipedia," *Semantic Web*, vol. 6, no. 2, pp. 167–195, 2015.

[6] J. Berant, A. Chou, R. Frostig, and P. Liang, "Semantic parsing on freebase from question-answer pairs," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2013, pp. 1533–1544.

[7] S. He, K. Liu, Y. Zhang, L. Xu, and J. Zhao, "Question answering over linked data using first-order logic," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2014, pp. 1092–1103.

[8] B. Hachey, W. Radford, J. Nothman, M. Honnibal, and J. R. Curran, "Evaluating entity linking with Wikipedia," *Artif. Intell.*, vol. 194, pp. 130–150, 2013.

[9] A. Bordes, N. Usunier, and A. Garcia-Durán, "Translating embeddings for modeling multi-relational data," in *Proc. 26th Int. Conf. Neural Inf. Process. Syst.*, 2013, pp. 2787–2795.

[10] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *Proc. 28th AAAI Conf. Artif. Intell.*, 2014, pp. 1112–1119.

[11] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, "Knowledge graph embedding via dynamic mapping matrix," in *Proc. 53rd Annu. Meet. Assoc. Comput. Linguistics 7th Int. Joint Conf. Natural Lang. Process.*, 2015, pp. 687–696.

[12] R. Socher, D. Chen, C. D. Manning, and A. Ng, "Reasoning with neural tensor networks for knowledge base completion," in *Proc. 26th Int. Conf. Neural Inf. Process. Syst.*, 2013, pp. 926–934.

[13] M. Nickel, V. Tresp, and H. P. Kriegel, "A three-way model for collective learning on multi-relational data," in *Proc. Int. Conf. Mach. Learn.*, 2011, pp. 809–816.

[14] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 2181–2187.

[15] R. Jenatton, N. L. Roux, A. Bordes, and G. Obozinski, "A latent factor model for highly multi-relational data," in *Proc. Int. Conf. Neural Inf. Process. Syst. 26th Annu. Conf. Neural Inf. Process. Syst.*, 2012, pp. 3176–3184.

[16] A. Bordes, X. Glorot, J. Weston, and Y. Bengio, "A semantic matching energy function for learning with multi-relational data," *Mach. Learn.*, vol. 94, no. 2, pp. 233–259, Feb. 2014.

[17] S. Schoenmackers, O. Etzioni, D. S. Weld, and J. Davis, "Learning first-order horn clauses from web text," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2010, pp. 1088–1098.

[18] N. Lao, T. M. Mitchell, and W. W. Cohen, "Random walk inference and learning in a large scale knowledge base," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2011, pp. 529–539.

[19] A. Neelakantan, B. Roth, and A. McCallum, "Compositional vector space models for knowledge base completion," in *Proc. 53rd Annu. Meet. Assoc. Comput. Linguistics 7th Int. Joint Conf. Natural Lang. Process. Asian Fed. Natural Lang. Process.*, 2015, pp. 156–166.

[20] A. McCallum, A. Neelakantan, R. Das, and D. Belanger, "Chains of reasoning over entities, relations, and text using recurrent neural networks," in *Proc. 15th Conf. Eur. Chapter Assoc. Comput. Linguistics*, 2017, pp. 132–141.

[21] K. Xu *et al.*, "Show, attend and tell: Neural image caption generation with visual attention," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 2048–2057.

[22] J. Mao, W. Xu, Y. Yang, J. Wang, and A. L. Yuille, "Deep captioning with multimodal recurrent neural networks (m-RNN)," in *Proc. Int. Conf. Learn. Representations*, 2014, vol. arXiv:abs/1412.6632.

[23] T. N. Sainath, O. Vinyals, A. W. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2015, pp. 4580–4584.

[24] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Int. Conf. Neural Inf. Process. Syst. 27th Annu. Conf. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.

[25] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proc. 31th Int. Conf. Mach. Learn.*, 2014, pp. 1188–1196.

[26] Y. Kim, "Convolutional neural networks for sentence classification," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2014, pp. 1746–1751.

[27] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," in *Proc. 52nd Annu. Meet. Assoc. Comput. Linguistics*, 2014, pp. 655–665.

[28] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *Proc. Int. Conf. Learn. Representations*, 2017, vol. arXiv:abs/1710.10903.

[29] J. S. Hochreiter, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[30] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Netw.*, vol. 18, no. 5/6, pp. 602–610, 2005.

[31] M. Fan, D. Zhao, Q. Zhou, Z. Liu, T. F. Zheng, and E. Y. Chang, "Distant supervision for relation extraction with matrix completion," in *Proc. 52nd Annu. Meet. Assoc. Comput. Linguistics*, 2014, pp. 839–849.

[32] X. Han, Z. Liu, and M. Sun, "Denoising distant supervision for relation extraction via instance-level adversarial training," *arXiv*, vol. abs/1805.10959, 2018.

[33] E. G. Dave Orr, A. Subramanya, and M. Ringgaard, "11 billion clues in 800 million documents: A web research corpus annotated with freebase concepts," 2013. [Online]. Available: http://googleresearch.blogspot.com/2013/07/11-billion-clues-in-800-million.html

[34] Y. Lin, Z. Liu, X. Zhu, X. Zhu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 2181–2187.

[35] B. Yang, W. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," in *Proc. Int. Conf. Learn. Representations*, May 2015, vol. arXiv:abs/1412.6575.

[36] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *Proc. 33nd Int. Conf. Mach. Learn.*, 2016, pp. 2071–2080.

[37] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge, U.K.: Cambridge Univ. Press, 2008.

[38] B. Yang, W. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," in *Proc. Int. Conf. Learn. Representations*, May 2015, vol. arXiv:abs/1412.6575.

[39] N. Lao, A. Subramanya, F. Pereira, and W. W. Cohen, "Reading the web with learned syntactic-semantic inference rules," in *Proc. Joint Conf. Empirical Methods Natural Lang. Process. Comput. Natural Lang. Learn.*, 2012, pp. 1017–1026.

[40] M. Gardner, P. P. Talukdar, B. Kisiel, and T. Mitchell, "Improving learning and inference in a large knowledge-base using latent syntactic cues," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2013, pp. 833–838.

[41] K. Guu, J. Miller, and P. Liang, "Traversing knowledge graphs in vector space," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 318–327.

[42] K. Toutanova, V. Lin, W. Yih, H. Poon, and C. Quirk, "Compositional learning of embeddings for relation paths in knowledge base and text," in *Proc. 54th Annu. Meet. Assoc. Comput. Linguistics*, 2016, pp. 1434–1444.

[43] T. Trouillon, C. R. Dance, É. Gaussier, J. Welbl, S. Riedel, and G. Bouchard, “Knowledge graph completion via complex tensor factorization,” *J. Mach. Learn. Res.*, vol. 18, pp. 130:1–130:38, 2017.

[44] H. Xiao, M. Huang, and X. Zhu, “TransG: A generative model for knowledge graph embedding,” in *Proc. 54th Annu. Meet. Assoc. Comput. Linguistics*, 2016, pp. 2316–2325.

[45] Q. Xie, X. Ma, Z. Dai, and E. Hovy, “An interpretable knowledge transfer model for knowledge base completion,” in *Proc. Meet. Assoc. Comput. Linguistics*, 2017, pp. 950–962.

[46] L. Vilnis and A. Mccallum, “Word representations via gaussian embedding,” in *Proc. Int. Conf. Learn. Representations*, 2014, vol. abs/1412.6623.

[47] A. Bordes, X. Glorot, J. Weston, and Y. Bengio, “Learning structured embeddings of knowledge bases,” in *Proc. 25th AAAI Conf. Artif. Intell.*, 2011, pp. 301–306.

[48] G. Ji, K. Liu, S. He, and J. Zhao, “Knowledge graph completion with adaptive sparse transfer matrix,” in *Proc. 13th AAAI Conf. Artif. Intell.*, 2016, pp. 985–991.

[49] M. Fan, Q. Zhou, E. Chang, and T. F. Zheng, “Transition-based knowledge graph embedding with relational mapping properties,” in *Proc. 28th Pacific Asia Conf. Lang. Inf. Comput.*, 2014, pp. 328–337.

[50] J. Feng, M. Huang, M. Wang, M. Zhou, Y. Hao, and X. Zhu, “Knowledge graph embedding by flexible translation,” in *Proc. 15th Int. Conf. Princ. Knowl. Representation Reasoning*, 2016, pp. 557–560.

[51] H. Xiao, M. Huang, Y. Hao, and X. Zhu, “TransA: An adaptive approach for knowledge graph embedding,” *arXiv*, vol. abs/1509.05490, 2015.

[52] A. Garc ía-Durán, A. Bordes, and N. Usunier, “Effective blending of two and three-way interactions for modeling multi-relational data,” in *Proc. Eur. Conf. Mach. Learn. Knowl. Discovery Databases*, 2014, pp. 434–449.

[53] T. Trouillon and M. Nickel, “Complex and holographic embeddings of knowledge graphs: A comparison,” *arXiv*, vol. abs/1707.01475, 2017.

[54] H. Liu, Y. Wu, and Y. Yang, “Analogical inference for multi-relational embeddings,” in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 2168–2178.

[55] X. Zhou, Q. Zhu, P. Liu, and L. Guo, “Learning knowledge embeddings by combing limit-based scoring loss,” in *Proc. 26th ACM Conf. Inf. Knowl. Manage.*, 2017, pp. 1009–1018.

Qiannan Zhu received the MS degree in the Institute of Information Engineering, Chinese Academy of Sciences, in 2018. She is currently working toward the PhD degree at the Institute of Information Engineering, Chinese Academy of Sciences. Her current research interests include knowledge graph representation, recommendation system and natural language process.

Xiaofei Zhou received the PhD degree in Pattern Recognition and Artificial Intelligence from Nanjing University of Science and Technology, in 2008. She as a Postdoctor worked in Graduate University of Chinese Academy of Sciences, from 2008 to 2010. She is currently an associate professor in the Institute of Information Engineering, Chinese Academy of Sciences. Her research interests include machine learning, natural language processing and knowledge graph.

Jianlong Tan received the PhD degree from the Institute of Computing Technology, Chinese Academy of Sciences. He is currently a professor with the Institute of Information Engineering, Chinese Academy of Sciences. His research interests include data stream management systems and information security.

Li Guo is a professor with the Institute of Information Engineering, Chinese Academy of Sciences. She is also the chairman with the Intelligent Information Processing Research Center, Institute of Information Engineering, Chinese Academy of Sciences. Her research interests include data stream management systems and information security.