Learning to Exploit Long-term Relational Dependencies in Knowledge Graphs

Lingbing Guo [1] **Zequn Sun** [1] **Wei Hu** [1]

Abstract

We study the problem of knowledge graph (KG) embedding. A widely-established assumption to this problem is that similar entities are likely to have similar relational roles. However, existing related methods derive KG embeddings mainly based on triple-level learning, which lack the capability of capturing long-term relational dependencies of entities. Moreover, triple-level learning is insufficient for the propagation of semantic information among entities, especially for the case of cross-KG embedding. In this paper, we propose recurrent skipping networks (RSNs), which employ a skipping mechanism to bridge the gaps between entities. RSNs integrate recurrent neural networks (RNNs) with residual learning to efficiently capture the long-term relational dependencies within and between KGs. We design an end-to-end framework to support RSNs on different tasks. Our experimental results showed that RSNs outperformed state-of-the-art embedding-based methods for entity alignment and achieved competitive performance for KG completion.

1. Introduction

Knowledge graphs (KGs) store a wealth of structured facts about the real world. Each fact is structured in the form of (s, r, o), where s, o and r denote the subject entity, object entity and their relation, respectively. KGs have gradually become an important resource for many knowledge-driven applications, such as semantic search, question answering and recommender systems. Oftentimes, a single KG is far from complete and cannot support these applications with sufficient facts. To address this problem, two fundamental KG tasks are proposed: (i) **entity alignment**, a.k.a. entity resolution or matching, which aims at integrating multiple KGs by identifying entities in different KGs referring to the same real-world object; and (ii) **KG completion**, a.k.a. link prediction, which aims to complete the missing facts in a single KG. Conventional methods usually rank candidates by exploiting various features, as well as using crowdsourcing (Lacoste-Julien et al., 2013; Suchanek et al., 2012; Zhuang et al., 2017). However, even for a single KG, it can be developed and maintained by different people using different domain knowledge and natural languages, which inevitably makes it heterogeneous. Recently, several methods leverage KG embedding techniques to tackle this problem (Bordes et al., 2013; Dettmers et al., 2018; Chen et al., 2017; Zhu et al., 2017; Sun et al., 2018). They have shown effectiveness in learning relational information either in a single KG or across multiple KGs.

For KG embedding, existing methods start with the assumption that similar entities are likely to have similar relational roles. Their primary focus, therefore, lies in learning from relational triples of entities. Typically, some of them are inspired by the TransE model (Bordes et al., 2013), which interprets (s, r, o) as $\mathbf{s} + \mathbf{r} \approx \mathbf{o}$, where the boldfaces denote the corresponding embeddings. Under this modeling, the embedding of one entity is learned by aggregating the embeddings of its 1-hop relational neighbors. We refer to this kind of models as *triple-level learning*. Many KG embedding models belong to this kind, including not only translational models like TransE (Bordes et al., 2013), TransH (Wang et al., 2014) and TransR (Lin et al., 2015b), but also compositional models like DistMult (Yang et al., 2015), ComplEx (Trouillon et al., 2016) and HolE (Nickel et al., 2016), as well as neural models like ProjE (Shi & Weninger, 2017) and ConvE (Dettmers et al., 2018).

Triple-level learning has two major limitations: (i) **low expressiveness**. It learns entity embeddings from a fairly local view (i.e., 1-hop relational neighbors). On one hand, there are many different entities having common local relational neighbors in KGs, such as entities with multi-mapping relations as discussed in (Wang et al., 2014). Exploiting local relational neighbors for KG embedding is insufficient. On the other hand, there are many entities having few relational triples (a.k.a. long-tail entities) in real-world KGs (Li et al., 2017). With triple-level learning, long-tail entities would receive limited attention, thus their embeddings have low expressiveness; and (ii) **inefficient information propagation**. For the entity alignment task, existing methods rely on *seed*

[1]State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, Jiangsu, China. Correspondence to: Wei Hu <whu@nju.edu.cn>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

alignment (i.e., prior entity alignment known ahead of time) to bridge two KGs. As triple-level learning uses relational triples of seed entities (entities in seed alignment) to deliver alignment information across KGs, it would limit alignment propagation, especially for long-tail entities and entities that are far away from seed entities. Although the information of multi-hop neighborhoods can be passed with back propagation in different mini-batches (Wang et al., 2017), the efficiency would be seriously affected, especially in the case of cross-KG embedding.

To deal with the limitations, we propose *recurrent skipping networks* (RSNs). Instead of learning the embeddings in a triple-level view, RSNs concentrate on learning from relational paths. A relational path is defined as an entity-relation chain, such as (*United Kingdom*, *country*$^-$, *Tim Berners-Lee*, *employer*, *W3C*), where *country*$^-$ is a reverse relation that we create additionally to enhance the connectivity. It is clear that paths can provide richer relational dependencies than triples without losing the local relational information of entities. RSNs also overcome the limitations that many existing methods are only designed for one specific task of KG embedding. For example, TransR (Lin et al., 2015b) and ConvE (Dettmers et al., 2018) have competitive performance on the KG completion task, but they fail on the entity alignment task. We explain the reasons in later sections.

A conventional choice to model relational paths is recurrent neural networks (RNNs). However, RNNs assume that the next element in a sequence depends on the current input and the previous hidden state only, which is inappropriate for KG path modeling. Take a relational path $(..., s, r, o, ...)$ for example. After being fed with $(..., \mathbf{s}, \mathbf{r})$, RNNs use the current input $\mathbf{r}$ and the previous hidden state $\mathbf{h}_s$ to infer o. However, $\mathbf{h}_s$ is a mix of context, which overlooks the importance of $\mathbf{s}$. In KGs, subject entities are vital for inferring a specific object entity. The local neighbor information would be broken if we use RNNs to model relational paths. To overcome this weakness, RSNs enable the output hidden states of relations to learn a *residual* (He et al., 2016) from their direct subject entities when inferring object entities, with only a few more parameters.

Furthermore, we present an end-to-end framework to support RSNs on different tasks. Specifically, to obtain desired paths, we use the *biased random walks* to efficiently sample paths from KGs. This sampling method differs from normal random walks in that it can fluently control the depth and cross-KG biases of the generated paths. After sampling the paths, we are capable of using RSNs to model the relational paths. To make the embedding learning more effectively, we design *type-based noise-constrained estimation* (NCE), which optimizes the negative example sampling according to the types of elements in paths.

The main contributions of this paper are listed as follows:

- We propose the path-level learning for KG embedding and design RSNs to remedy the limitations of using RNNs to model relational paths. (Section 3)
- We present an end-to-end framework to support different KG embedding tasks. It significantly outperformed several state-of-the-art methods for entity alignment and achieved competitive performance for KG completion. (Sections 4 and 5)

2. Related Work

2.1. Path-level Embedding

PTransE (Lin et al., 2015a) is one of the path-based KG embedding models. It improves TransE (Bordes et al., 2013) by incorporating relation inferences into KG embedding. For example, if there exist a path $(e_1, r_1, e_2, r_2, e_3)$ and a triple (e_1, r_3, e_3), PTransE models a relation inference by learning $\mathbf{r_1} \oplus \mathbf{r_2} \approx \mathbf{r_3}$, where $\oplus$ denotes the used operator, e.g., add, to merge $\mathbf{r_1}$ and $\mathbf{r_2}$. However, it is worth noting that PTransE only uses relation sequences to enhance triple-level learning but ignores relational dependencies of entities. Thus, PTransE still belongs to the triple-level learning. There are many similar methods that purely leverage relational paths or employ chunk-based paths (Guu et al., 2015; McCallum et al., 2017; Yang et al., 2017). Different from them, our approach is the first one to fully exploit the potential of KG paths.

In the network embedding area, DeepWalk (Perozzi et al., 2014) uses the uniform random walks to sample paths in networks and employs Skip-Gram (Mikolov et al., 2013) to model these paths. Skip-Gram learns node embeddings by maximizing the probabilities of their neighbors. node2vec (Grover & Leskovec, 2016) introduces the biased random walks to refine the process of path sampling from networks. It smoothly controls the node selection strategy to make the random walks explore neighbors in a breadth-first-search as well as a depth-first-search fashion. In this paper, the proposed biased random walks are inspired by node2vec. However, we concentrate on generating deep and cross-KG paths. There are also many methods for graph embedding, e.g., structure2vec (Dai et al., 2016), SSE (Dai et al., 2018), and JK-Net(Xu et al., 2018). Similar to the network embedding models, they usually do not consider the semantics and directions of relations. Their main goal is to discover clusters or communities of related nodes. Therefore, we think that these methods cannot directly model complex and directed relations in KGs.

2.2. KG Embedding

KG embedding has been widely studied in last few years. TransE (Bordes et al., 2013) presents translational embedding, which models a relational triple (s, r, o) as $\mathbf{s} + \mathbf{r} \approx \mathbf{o}$.

TransH (Wang et al., 2014) and TransR (Lin et al., 2015b) improve TransE on modeling complex relations. There are also many non-translational methods. ComplEx (Trouillon et al., 2016) embeds KGs into complex spaces to enhance the basic model DistMult (Yang et al., 2015). RotatE (Sun et al., 2019) is similar to ComplEx, but it defines each relation as a rotation from the subject entity to the object entity. Recently, there also exist several neural models designed for KG completion. ProjE (Shi & Weninger, 2017) adopts a simple but effective shared variable neural network, and achieves competitive performance. ConvE (Dettmers et al., 2018) combines the embeddings of subject entities and relations by a 2D convolutional operation. For more KG completion methods, please see (Wang et al., 2017).

Recently, several studies (Chen et al., 2017; Sun et al., 2017; 2018; Chen et al., 2018) have found that KG embedding can also improve the performance on the entity alignment task. MTransE (Chen et al., 2017) reuses TransE (Bordes et al., 2013) to separately train embeddings of different KGs and learns a transition between the KG embeddings. JAPE (Sun et al., 2017) is also based on TransE, but it learns embeddings of different KGs in a unified space. Additionally, it leverages attributes to refine entity embeddings. IPTransE (Zhu et al., 2017) employs an iterative alignment process to extend PTransE (Lin et al., 2015a) for entity alignment. As aforementioned, it still belongs to the triple-level learning. BootEA (Sun et al., 2018) bootstraps embedding-based entity alignment by using an elaborate algorithm to update alignment during iterations. KDCoE (Chen et al., 2018) co-trains entity relations and descriptions to derive KG embeddings. It requires extra pre-trained multi-lingual word embeddings and descriptions. All these methods use TransE-like models to learn KG embeddings, thus they are not capable of capturing long-term relational dependencies in KGs and the propagation of alignment information between different KGs is also limited. GCN-Align (Wang et al., 2018) employs graph convolutional networks (GCNs) to embed entities based on adjacent neighborhoods, but it does not consider relation semantics among entities.

3. Recurrent Skipping Networks

In this section, we start with preliminaries and an introduction to RNNs. Then, we describe RSNs in detail. Finally, we compare RSNs with conventional residual learning.

3.1. Preliminaries

A KG is a directed multi-relational graph where nodes denote entities and edges have labels indicating that there exist some specific relations between the connected entities. Formally, we define a KG as a 3-tuple $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$, where $\mathcal{E}$ and $\mathcal{R}$ denote the sets of entities and relations, respectively. $\mathcal{T} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ is the set of relational triples.

Different from the existing methods that learn from triples, in this paper, we concentrate on learning from relational paths. A relational path is an entity-relation chain, where entities and relations appear alternately. The head and tail of a relational path must be entities. We use $(x_1, x_2, ..., x_T)$ to denote a relational path, where T is an odd number. Elements with odd indices are entities while the remaining is intermediate relations. To enhance the connectivity of KGs, we add reverse relations in KGs. For each triple (s, r, o), we add a reverse triple (o, r^-, s), where r^- is distinct from r.

KG completion is a prevalent task for KG embedding (Bordes et al., 2013). Given a KG, it aims to predict the object entity o given $(s, r, ?)$ or predict the subject entity s given $(?, r, o)$.

Given two KGs $\mathcal{G}_1 = (\mathcal{E}_1, \mathcal{R}_1, \mathcal{T}_1)$ and $\mathcal{G}_2 = (\mathcal{E}_2, \mathcal{R}_2, \mathcal{T}_2)$, entity alignment aims to find aligned entity pairs between them. Typically, a small subset of entity alignment, denoted by $\mathcal{S} \subset \mathcal{E}_1 \times \mathcal{E}_2$, is known as seed alignment. So, the input of entity alignment is $\mathcal{G}_1, \mathcal{G}_2$ and $\mathcal{S}$. Oftentimes, the two KGs are assembled as one *joint* KG by copying relational triples of seed entities to their counterparts. For convenience, we also denote the joint KG by $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$.

3.2. RNNs

RNNs are a popular class of neural networks performing well on sequential data types. Given a relational path $(x_1, x_2, ..., x_T)$ as input, we first convert the entities and relations into fixed d-dimensional embeddings. Thus, the relational path turns to an embedding sequence $(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T)$. RNNs sequentially read in elements in this sequence and output a hidden state at each time step. The output hidden state at time step t, denoted by $\mathbf{h}_t$, is calculated as follows:

$$\mathbf{h}_t = \tanh(\mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{W}_x \mathbf{x}_t + \mathbf{b}), \tag{1}$$

where $\mathbf{W}_h, \mathbf{W}_x$ are the weight matrices. $\mathbf{b}$ is the bias.

RNNs are capable of handling input of any length with a few parameters and have achieved state-of-the-art performance in many areas. However, there still exist a few limitations when using RNNs to model relational paths. First, the elements in a relational path have two different types: "entity" and "relation", which always appear in an alternating order. However, the traditional RNNs treat them as the same type like words or graph nodes, which makes capturing semantic information in relational paths less effective.

Second, any relational paths are constituted by triples, but these basic structure units are overlooked by RNNs. Let x_t denote a relation in a relational path and (x_{t-1}, x_t, x_{t+1}) denote a triple involving x_t. As shown in Eq. (1), to predict x_{t+1}, RNNs would combine the hidden state $\mathbf{h}_{t-1}$ and the current input $\mathbf{x}_t$, where $\mathbf{h}_{t-1}$ is a mix of the information of all the previous elements $x_1, ..., x_{t-1}$. In fact, it is expected

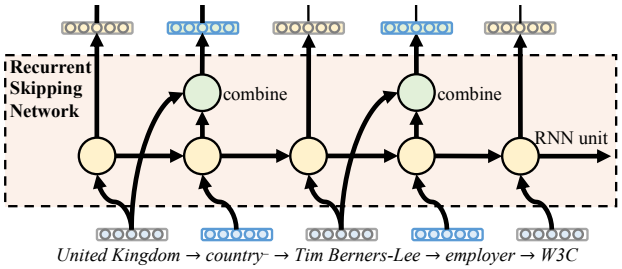


Figure 1. Example of RSNs with a 2-hop relational path

that the information of x_{t-1}, x_t in the current triple can be more emphasized.

3.3. Semantic Enhancement with Skipping Mechanism

To remedy the limitations of conventional RNNs, we propose RSNs, which refine RNNs by a simple but effective skipping mechanism. The basic idea of RSNs is to shortcut the current input entity to let it directly participate in predicting its object entity. In other words, an input element in a relational path whose type is "entity" can not only contribute to predicting its next relation, but also straightly take part in predicting its object entity. Figure 1 illustrates an RSN example.

Given a relational path $(x_1, x_2, ..., x_T)$, the skipping operation for an RSN is formulated as follows:

$$\mathbf{h}'_t = \begin{cases} \mathbf{h}_t & x_t \in \mathcal{E} \\ \mathbf{S}_1\mathbf{h}_t + \mathbf{S}_2\mathbf{x}_{t-1} & x_t \in \mathcal{R} \end{cases}, \tag{2}$$

where $\mathbf{h}'_t$ denotes the output hidden state of the RSN at time step t, and $\mathbf{h}_t$ denotes the corresponding RNN output. $\mathbf{S}_1, \mathbf{S}_2$ are the weight matrices, and we share their parameters at different time steps. In this paper, we choose the weighted sum for the skipping operation, but other combination methods can be employed as well.

3.4. Insight of RSNs

Intuitively, RSNs explicitly distinguish entities and relations, and allow subject entities to skip their connections for directly participating in predicting object entities. Behind this simple skipping operation, there is an important thought to adopt residual learning.

Let $F(\mathbf{x})$ be an original mapping, where $\mathbf{x}$ denotes the input, and $H(\mathbf{x})$ be the expected mapping. Compared to directly optimizing $F(\mathbf{x})$ to fit $H(\mathbf{x})$, conventional residual learning hypothesizes that it can be easier to optimize $F(\mathbf{x})$ to fit the residual part $H(\mathbf{x}) - \mathbf{x}$. For an extreme case, if an identity mapping is optimal (i.e., $H(\mathbf{x}) = \mathbf{x}$), pushing the residual to 0 would be much easier than fitting an identity mapping by a stack of nonlinear layers (He et al., 2016).

However, different from ResNet (He et al., 2016) and recurrent residual networks (RRNs) (Wang & Tian, 2016), which are proposed to help train very deep networks, RSNs employ residual learning on "shallow" networks. The skipping connections do not link the previous input to the very deep layers, but only focus on each triple in relational paths.

Table 1. Differences of RNNs, RRNs and RSNs, by an example (*United Kingdom, country⁻, Tim Berners-Lee, **employer**, W3C*)

Models	Optimize $F([\cdot], \textit{employer})$ as
RNNs	$F([\cdot], \textit{employer}) := \textit{W3C}$
RRNs	$F([\cdot], \textit{employer}) := \textit{W3C} - [\cdot]$
RSNs	$F([\cdot], \textit{employer}) := \textit{W3C} - \textit{Tim Berners-Lee}$

$[\cdot]$ denotes context (*United Kingdom, country⁻, Tim Berners-Lee*)

Specifically, given a relational path $(..., x_{t-1}, x_t, x_{t+1}, ...)$, where (x_{t-1}, x_t, x_{t+1}) forms a triple, RRNs leverage residual learning by regarding the process at each time step as a mini-residual network. Take time step t for example. RRNs take the previous hidden state $\mathbf{h}_{t-1}$ as input and learn the residual $\mathbf{h}_t$ by $H(\mathbf{h}_{t-1}, \mathbf{x}_t) - \mathbf{h}_{t-1}$, where $H(\mathbf{h}_{t-1}, \mathbf{x}_t)$ is the expected mapping for $\mathbf{h}_{t-1}, \mathbf{x}_t$. Since the information of x_{t-1} is mixed in $\mathbf{h}_{t-1}$, RRNs still ignore the structure of KGs that x_{t-1}, x_t should be more emphasized for predicting x_{t+1}. Hence, the local (i.e., 1-hop) relations cannot be appropriately modeled.

Differently, RSNs leverage residual learning in a new manner. Instead of choosing $\mathbf{h}_{t-1}$ as subtrahend, RSNs directly pick up the subject entity $\mathbf{x}_{t-1}$ as subtrahend. We can write this residual as follows:

$$\mathbf{h}_t := H(\mathbf{h}_{t-1}, \mathbf{x}_t) - \mathbf{x}_{t-1}, \quad x_t \in \mathcal{R}. \tag{3}$$

The underlying thought is that making the output hidden state $\mathbf{h}_t$ to fit $\mathbf{x}_{t+1}$ may be hard, but learning the residual of $\mathbf{x}_{t+1}$ and $\mathbf{x}_{t-1}$ may be easier. We think that this is the key characteristic of RSNs.

Table 1 shows the differences of RNNs, RRNs and RSNs by an example. Suppose that we are standing at *employer*, it is obvious that learning the residual between *W3C* and *Tim Berners-Lee* can make the optimization much easier. The skipping operation only increases a few more parameters, but it offers an efficient way to remedy the major problem of leveraging sequence models to learn relational paths. We also empirically demonstrate the strengths of RSNs in the performance and convergence speed in our experiments.

4. Architecture of RSNs

In this section, we present an end-to-end framework that leverages RSNs for entity alignment and KG completion. We show the full architecture in Appendix A.1. Three main modules in this framework are described as follows:

- **Biased random walk sampling** generates deep and cross-KG relational paths.

- **Recurrent skipping network** models relational paths to learn KG embeddings. We have introduced it in the previous section.
- **Type-based noise contrastive estimation** evaluates the loss of RSNs in an optimized way.

4.1. Biased Random Walks

Towards KG embedding, the desired relational paths should be relatively deep and, for entity alignment, stretch across two KGs. Deep paths carry more relational dependencies than triples for representing the relational roles of entities. Cross-KG paths serve as the bridges between two KGs to deliver alignment information.

Because KGs are often large scale, it is impractical to enumerate all possible paths. Besides, not all paths contribute to KG embeddings. Thus, we propose a path sampling method with biased random walks on a single KG and across two KGs, which can efficiently explore deep and cross-KG relational paths for embedding learning.

Conventional random walks. Using random walks to sample paths from networks has been widely studied for a long time (Perozzi et al., 2014). When being applied to KGs, the unbiased random walks obtain the probability distribution of next entities by the following equation:

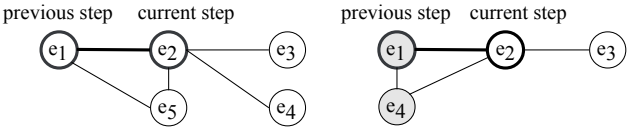
$$\Pr(e_{i+1} \mid e_i) = \begin{cases} \frac{\pi_{e_i \to e_{i+1}}}{Z} & \exists r \in \mathcal{R} \colon (e_i, r, e_{i+1}) \in \mathcal{T} \\ 0 & \text{otherwise} \end{cases}, \tag{4}$$

where e_i denotes the i^{th} entity in this walk. $\pi_{e_i \to e_{i+1}}$ is the unnormalized transition probability between e_i and e_{i+1}. Z is the normalization constant. The unbiased random walks choose next entities in a uniform probability distribution.

Biased random walks. We leverage the idea of second-order random walks (Grover & Leskovec, 2016) and introduce a *depth bias* to smoothly control the depths of sampled paths. Specifically, suppose that we are standing at entity e_i at present and the previous step is at e_{i-1}. The 1-hop neighbors of e_i are the candidates for the next step. As we prefer deep paths, we are inclined to choose the next entity which is far away from e_{i-1}. Formally, let e_{i+1} denote a candidate entity. We calculate the depth bias between e_{i-1} and e_{i+1}, denoted by $\mu_d(e_{i-1}, e_{i+1})$, as follows:

$$\mu_d(e_{i-1}, e_{i+1}) = \begin{cases} \alpha & d(e_{i-1}, e_{i+1}) = 2 \\ 1-\alpha & d(e_{i-1}, e_{i+1}) < 2 \end{cases}, \tag{5}$$

where $d(e_{i-1}, e_{i+1})$ gains the distance of the shortest path from e_{i-1} to e_{i+1}, and its values can only range in $\{0, 1, 2\}$. $\alpha \in (0, 1)$ is a hyper-parameter controlling the depths of random walks. To reflect the favors on deeper paths, we set $\alpha > 0.5$. Figure 2(a) illustrates an example of the depth-biased random walks. Candidates for the next step are e_3, e_4 and e_5. Their depth biases are as follows: $\mu_d(e_1, e_3) = \alpha$, $\mu_d(e_1, e_4) = \alpha$ and $\mu_d(e_1, e_5) = 1 - \alpha$. Due to $\alpha > 0.5$, we are more likely to go to e_3 or e_4.



(a) Depth-biased random walk (b) KG-biased random walk

Figure 2. Samples of biased random walks. For simplicity, we reduce a KG as an undirected graph by merging relations and their corresponding reversed ones. e_2 is the current entity that we now stand on and e_1 is the previous one.

Furthermore, we also encourage walking across two KGs to deliver alignment information for entity alignment. In a similar way, we introduce a *cross-KG bias* to favor paths connecting two KGs. To formalize, the cross-KG bias between e_{i-1} and e_{i+1}, denoted by $\mu_c(e_{i-1}, e_{i+1})$, is defined as follows:

$$\mu_c(e_{i-1}, e_{i+1}) = \begin{cases} \beta & kg(e_{i-1}) \neq kg(e_{i+1}) \\ 1-\beta & \text{otherwise} \end{cases}, \tag{6}$$

where $kg(\cdot)$ denotes the KG to which an entity belongs. $\beta \in (0, 1)$ is a hyper-parameter controlling the behavior of random walks across two KGs. To favor cross-KG paths, we set $\beta > 0.5$. This bias also avoids walking backwards and forwards between entities in the seed alignment. Let us look at Figure 2(b) as an example of KG-biased random walks. e_1 and e_4 are two entities in KG_1, while e_2 and e_3 are two entities in KG_2. e_2 is a seed entity. After walking from e_1 to e_2, we calculate the cross-KG biases as follows: $\mu_c(e_1, e_3) = \beta$ and $\mu_c(e_1, e_4) = 1 - \beta$. Due to $\beta > 0.5$, we prefer to go to e_3.

Finally, we combine the depth and cross-KG biases into the following bias:

$$\mu(e_{i-1}, e_{i+1}) = \mu_d(e_{i-1}, e_{i+1}) \times \mu_c(e_{i-1}, e_{i+1}). \tag{7}$$

The detailed algorithm of the biased random walk sampling is shown in Appendix A.2. Note that, biased random walks aim to sample paths which can properly describe a graph, rather than conditionally rank paths. Thus, it is significantly different from path ranking (Lao et al., 2011), which tends to select the paths with similar features due to their high rewards. In our case, we need *randomness* to ensure that all features of a graph are sampled.

4.2. Type-based NCE

Each element in a relational path can be optimized by learning to predict the next element. As the number of candidate

entities or relations is usually large, directly computing the sigmoid loss of each prediction is time-consuming. Thus, we use the noise-contrastive estimation (NCE) (Gutmann & Hyvärinen, 2010) to evaluate each output of RSNs, which only requires a small number of negative samples to approximate the integral distribution. To formalize, given the input $(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T)$, the loss of RSNs is defined as follows:

$$\mathcal{L} = -\sum_{t=1}^{T-1} \Big(\log \sigma(\mathbf{h}'_t \cdot \mathbf{y}_t) + \sum_{j=1}^{k} \mathbb{E}_{\tilde{y}_j \sim Q(\tilde{y})} \big[\log \sigma(-\mathbf{h}'_t \cdot \tilde{\mathbf{y}}_j) \big] \Big), \tag{8}$$

where $\mathbf{y}_t$ is the target at time step t, $\sigma(\cdot)$ is the sigmoid function, and k is the number of negative samples. A negative example $\tilde{y}_j$ is drawn from the noise probability distribution: $Q(\tilde{y}) \propto q(\tilde{y})^{\frac{3}{4}}$, where $q(\tilde{y})$ is the frequency of $\tilde{y}$ appearing in KGs.

Note that the negative samples can be either negative entities or negative relations based on the inference task (entity or relation prediction) at current step. So, we can separate the computation of noise probability distribution according to the target *types*. Specifically, if the current target is an entity, we draw negative samples from the noise probability distribution of entities. Negative relation sampling is carried out similarly. In this way, the candidate sets for negative sampling are compacted and the inapplicable negative examples can also be avoided.

5. Experiments and Results

We evaluated RSNs on two representative KG embedding tasks: entity alignment and KG completion. For each task, we conducted experiments on a set of real-world datasets and reported the results compared with several state-of-the-art methods. Due to lack of space, a part of experiments and results are shown in Appendix C.

5.1. Dataset Preparation

Entity alignment datasets. Although the existing datasets used by the embedding-based entity alignment methods (Chen et al., 2017; Sun et al., 2017; 2018; Wang et al., 2018) are sampled from real-world KGs, e.g., DBpedia and Wikidata, their entity distributions are quite different from real ones. We argue that this distortion would prevent us from a comprehensive and accurate evaluation of embedding-based entity alignment. In this paper, we design a *segment-based random PageRank sampling* (SRPRS) method, which can fluently control the degree distributions of entities in the sampled datasets. Here, the degree of an entity is defined as the number of relational triples in which the entity involves. We obtained four couples of datasets for embedding-based entity alignment, and each has a normal entity distribution and a dense one. Please see Appendix B for more details.

KG completion datasets. We considered two benchmark datasets, namely FB15K and WN18, for KG completion (Bordes et al., 2013). FB15K contains 15,000 entities, while WN18 has 18 types of relations. Furthermore, recent studies (Toutanova & Chen, 2015; Dettmers et al., 2018) argued that the two datasets contain redundant triples between the training and test sets. In Appendix C.1, we also showed the results on a modified version called FB15K-237.

5.2. Experiment Settings

We implemented RSNs with TensorFlow. The source code and datasets are accessible online.[1] Please see Appendix A.3 for the implementation details. We chose Hits@1, Hits@10 and mean reciprocal rank (MRR) as the evaluation metrics.

For entity alignment, we picked up several state-of-the-art embedding-based methods for comparison: MTransE (Chen et al., 2017), IPTransE (Zhu et al., 2017), JAPE (Sun et al., 2017), BootEA (Sun et al., 2018) and GCN-Align (Wang et al., 2018). As KDCoE (Chen et al., 2018) did not release its full code and we did not particularly sample entities with textual descriptions, we skipped this method. We also deployed the source code of a few KG completion methods on the joint KGs and considered them as additional baselines: TransR (Lin et al., 2015b), TransD (Ji et al., 2015), ConvE (Dettmers et al., 2018) and RotatE (Sun et al., 2019). Following the previous works, we used 30% of reference alignment as the seed alignment. We tried our best to tune the hyper-parameters for all the methods.

For KG completion, we mainly reused the results reported in literature. Due to a few methods did not report the results of some metrics, we conducted the experiments by using the provided source code. Following the previous works, we used the filtered ranks, which means that we would exclude other correct entities when we rank the current test entity.

5.3. Entity Alignment Results

Tables 2 and 3 depict the entity alignment results on the normal and dense datasets, respectively. It is evident that capturing long-term dependencies by relational paths enabled RSNs to outperform all the existing embedding-based entity alignment methods. Also, RSNs achieved better results than RSNs (w/o biases), which demonstrated the effectiveness of the proposed biased random walks.

Intuitively, the heterogeneity among different KGs is more severe than one KG with different languages. Therefore, entity alignment between different KGs is harder for the embedding-based entity alignment methods. By establish-

[1] https://github.com/nju-websoft/RSN

Table 2. Entity alignment results on the normal datasets

Methods	DBP-WD			DBP-YG			EN-FR			EN-DE		
	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR
MTransE	22.3	50.1	0.32	24.6	54.0	0.34	25.1	55.1	0.35	31.2	58.6	0.40
IPTransE	23.1	51.7	0.33	22.7	50.0	0.32	25.5	55.7	0.36	31.3	59.2	0.41
JAPE	21.9	50.1	0.31	23.3	52.7	0.33	25.6	56.2	0.36	32.0	59.9	0.41
BootEA	32.3	63.1	0.42	31.3	62.5	0.42	31.3	62.9	0.42	44.2	70.1	0.53
GCN-Align	17.7	37.8	0.25	19.3	41.5	0.27	15.5	34.5	0.22	25.3	46.4	0.33
TransR†	5.2	16.9	0.09	2.9	10.3	0.06	3.6	10.5	0.06	5.2	14.3	0.09
TransD†	27.7	57.2	0.37	17.3	41.6	0.26	21.1	47.9	0.30	24.4	50.0	0.33
ConvE†	5.7	16.0	0.09	11.3	29.1	0.18	9.4	24.4	0.15	0.8	9.6	0.03
RotatE†	17.2	43.2	0.26	15.9	40.1	0.24	14.5	39.1	0.23	31.9	55.0	0.40
RSNs (w/o biases)	37.2	63.5	0.46	36.5	62.8	0.45	32.4	58.6	0.42	45.7	69.2	0.54
RSNs	**38.8**	**65.7**	**0.49**	**40.0**	**67.5**	**0.50**	**34.7**	**63.1**	**0.44**	**48.7**	**72.0**	**0.57**

"†" denotes KG completion methods conducted with the source code on the joint KGs. The same to the following.

Table 3. Entity alignment results on the dense datasets

Methods	DBP-WD			DBP-YG			EN-FR			EN-DE		
	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR
MTransE	38.9	68.7	0.49	22.8	51.3	0.32	37.7	70.0	0.49	34.7	62.0	0.44
IPTransE	43.5	74.5	0.54	23.6	51.3	0.33	42.9	78.3	0.55	34.0	63.2	0.44
JAPE	39.3	70.5	0.50	26.8	57.3	0.37	40.7	72.7	0.52	37.5	66.1	0.47
BootEA	67.8	91.2	0.76	68.2	89.8	0.76	64.8	91.9	0.74	66.5	87.1	0.73
GCN-Align	43.1	71.3	0.53	31.3	57.5	0.40	37.3	70.9	0.49	32.1	55.2	0.40
TransR†	14.1	38.6	0.22	13.0	38.0	0.21	15.2	43.8	0.25	10.7	30.9	0.18
TransD†	60.5	86.3	0.69	62.1	85.2	0.70	54.9	86.0	0.66	57.9	81.6	0.66
ConvE†	30.8	50.5	0.38	37.2	57.0	0.44	30.0	49.7	0.37	42.3	60.3	0.49
RotatE†	62.2	86.5	0.71	65.0	87.2	0.73	48.6	80.4	0.59	63.2	83.2	0.70
RSNs (w/o biases)	74.6	90.8	0.80	80.2	95.0	0.86	73.2	90.7	0.80	71.0	87.2	0.77
RSNs	**76.3**	**92.4**	**0.83**	**82.6**	**95.8**	**0.87**	**75.6**	**92.5**	**0.82**	**73.9**	**89.0**	**0.79**

ing long-term dependencies, RSNs captured richer information of KGs and learned more accurate embeddings, leading to more significant improvement on the DBP-WD and DBP-YG datasets, especially on the normal datasets.

Our experimental results also showed that the embedding-based entity alignment methods are sensitive to entity distributions. The performance of all the methods on the normal datasets is significantly lower than that on the dense datasets, because the dense datasets contain richer relational triples for KG embedding. Although the normal datasets are more difficult, RSNs still gained considerable advantages compared with the other methods. This stemed from the fact that RSNs learn from relational paths, which can preserve more semantics than triples.

It is worth mentioning that RSNs showed larger superiority in terms of Hits@1 and MRR. Hits@1 only considers the correct results at the first position, while MRR also favors the top-ranked results. As aforementioned, RSNs can capture richer information to help identify aligned entities in different KGs. The better results on these two more important metrics verified this point.

5.4. KG Completion Results

We also conducted experiments to assess the performance of RSNs on KG completion, by deactivating the cross-KG bias in random walks. Specifically, subject entity s and relation r are regarded as a sequence of length 2. We fed their embeddings to RSNs to predict the next element (i.e., object entity o). The experimental results are shown in Tables 4 and 5. We can see that RSNs obtained comparable performance on both two datasets. More specifically, RotatE performed best on FB15K, followed by our RSNs, which also showed a clear advantage compared with the others. However, their performance gaps were significantly narrowed on WN18. It is worth noting that RSNs outperformed all the translational models that also aim to learn KG embeddings rather than only complete KGs.

5.5. Explanations of the Results

Entity alignment and KG completion exist significant divergences. Several methods that performed pretty well on KG completion, e.g., ConvE, lost their advantages on entity alignment. We argue that this may be caused by that they

Table 4. KG completion results on FB15K

Methods	Hits@1	Hits@10	MRR
TransE‡	30.5	73.7	0.46
TransR‡	37.7	76.7	0.52
TransD‡	31.5	69.1	0.44
ComplEx	59.9	84.0	0.69
ConvE	67.0	87.3	0.75
RotatE	**74.6**	**88.4**	**0.80**
RSNs (w/o cross-KG bias)	72.2	87.3	0.78

"‡" denotes methods executed by ourselves using the source code, due to certain metrics were not evaluated.

Table 5. KG completion results on WN18

Methods	Hits@1	Hits@10	MRR
TransE‡	27.4	94.4	0.58
TransR‡	54.8	94.7	0.73
TransD‡	30.1	93.1	0.56
ComplEx	93.6	94.7	0.94
ConvE	93.5	95.5	0.94
RotatE	**94.4**	**95.9**	**0.95**
RSNs (w/o cross-KG bias)	92.2	95.3	0.94

were particularly designed for KG completion. In other words, they aim to better model a triple instead of learning the relational dependencies in KGs. For instance, ConvE involves the convolutional operation to better predict the missing entities, but the complex networks may hinder the learning of input embeddings. But for entity alignment, we identify aligned entities by directly comparing the trained embeddings. These methods may not be capable of training high-quality embeddings.

We also found that RSNs performed better on entity alignment than KG completion. As aforementioned, the performance of KG completion can largely be improved with a sophisticatedly-designed structure for triples, whereas the main gaol of RSNs is to model the long paths. This limits the performance of RSNs for KG completion, which only needs to predict subject or object entities in triples.

6. Further Experiments

6.1. Comparison with Alternative Networks

To assess the feasibility of RSNs, we conducted experiments to compare with RNNs and RRNs (Wang & Tian, 2016). Both RNNs and RRNs used in this experiment were implemented with the same settings of multi-layer LSTM units, dropout and batch normalization.

We depict the comparison results on the DBP-WD dataset in Figure 3. Because RNNs and RRNs do not consider the local structures of relational paths, they converged at a very slow speed. Differently, RSNs achieved better performance with only $1/30$ epochs, which indicated that this particular residual structure is vital for learning relational paths in KGs. Furthermore, RRNs only achieved little improvement compared with RNNs. This implied that simply combining residual learning with RNNs did not significantly help.

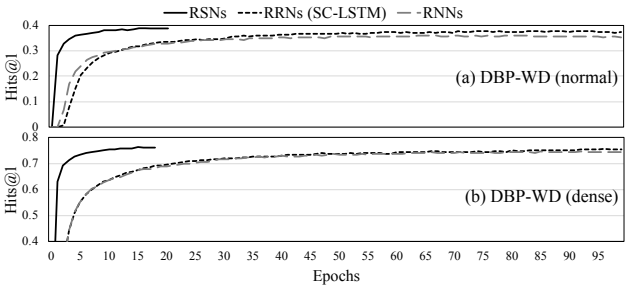
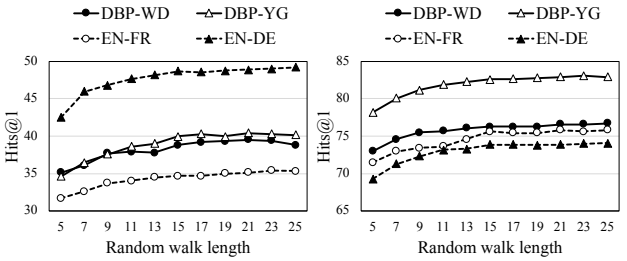


Figure 3. Hits@1 results w.r.t. epochs to converge



(a) Normal datasets (b) Dense datasets

Figure 4. Hits@1 results w.r.t. random walk length

6.2. Sensitivity to Random Walk Length

We also want to observe how the random walk length affects the performance of RSNs. In Figure 4, on all the eight entity alignment datasets, the Hits@1 results increase sharply from length 5 to 15, which indicates that modeling longer relational paths can help KG embedding obtain better performance. Also, we saw that the performance approaches to saturation from length 15 to 25, which may mean that RSNs have reached the max-length of capturing dependencies in the relational paths. In consideration of efficiency, the results in Tables 2 and 3 are based on length 15. More sensitivity analyses can be found in Appendix C.2.

7. Concluding Remarks

In this paper, we studied the path-level KG embedding learning and proposed RSNs to remedy the problems of using sequence models to learn relational paths. We presented an end-to-end framework, which uses the biased random walks to sample desired paths and models them with RSNs. Our experiments showed that the proposed method can obtain superior performance for entity alignment and competitive results for KG completion. Future work includes studying a unified sequence model to learn KG embeddings using both relational paths and textual information.

Acknowledgements

This work is supported by the National Key R&D Program of China (No. 2018YFB1004300), the National Natural Science Foundation of China (No. 61872172), and the Key R&D Program of Jiangsu Science and Technology Department (No. BE2018131).

References

Bordes, A., Usunier, N., Garcia-Durán, A., Weston, J., and Yakhnenko, O. Translating embeddings for modeling multi-relational data. In *NIPS*, pp. 2787–2795, 2013.

Chen, M., Tian, Y., Yang, M., and Zaniolo, C. Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. In *IJCAI*, pp. 1511–1517, 2017.

Chen, M., Tian, Y., Chang, K.-W., Skiena, S., and Zaniolo, C. Co-training embeddings of knowledge graphs and entity descriptions for cross-lingual entity alignment. In *IJCAI*, pp. 3998–4004, 2018.

Dai, H., Dai, B., and Song, L. Discriminative embeddings of latent variable models for structured data. In *ICML*, pp. 2702–2711, 2016.

Dai, H., Kozareva, Z., Dai, B., Smola, A., and Song, L. Learning steady-states of iterative algorithms over graphs. In *ICML*, pp. 1114–1122, 2018.

Dettmers, T., Minervini, P., Stenetorp, P., and Riedel, S. Convolutional 2D knowledge graph embeddings. In *AAAI*, pp. 1811–1818, 2018.

Grover, A. and Leskovec, J. node2vec: Scalable feature learning for networks. In *KDD*, pp. 855–864, 2016.

Gutmann, M. and Hyvärinen, A. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTAS*, pp. 297–304, 2010.

Guu, K., Miller, J., and Liang, P. Traversing knowledge graphs in vector space. In *EMNLP*, pp. 318–327, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, pp. 770–778, 2016.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pp. 448–456, 2015.

Ji, G., He, S., Xu, L., Liu, K., and Zhao, J. Knowledge graph embedding via dynamic mapping matrix. In *ACL*, pp. 687–696, 2015.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Lacoste-Julien, S., Palla, K., Davies, A., Kasneci, G., Graepel, T., and Ghahramani, Z. SIGMa: Simple greedy matching for aligning large knowledge bases. In *KDD*, pp. 572–580, 2013.

Lao, N., Mitchell, T., and Cohen, W. W. Random walk inference and learning in a large scale knowledge base. In *EMNLP*, pp. 529–539, 2011.

Leskovec, J. and Faloutsos, C. Sampling from large graphs. In *KDD*, pp. 631–636, 2006.

Li, F., Dong, X. L., Langen, A., and Li, Y. Knowledge verification for long-tail verticals. *PVLDB*, 10(11):1370–1381, 2017.

Lin, Y., Liu, Z., Luan, H.-B., Sun, M., Rao, S., and Liu, S. Modeling relation paths for representation learning of knowledge bases. In *EMNLP*, pp. 705–714, 2015a.

Lin, Y., Liu, Z., Sun, M., Liu, Y., and Zhu, X. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, pp. 2181–2187, 2015b.

McCallum, A., Neelakantan, A., Das, R., and Belanger, D. Chains of reasoning over entities, relations, and text using recurrent neural networks. In *EACL*, pp. 132–141, 2017.

Mikolov, T., Yih, W., and Zweig, G. Linguistic regularities in continuous space word representations. In *NAACL-HLT*, pp. 746–751, 2013.

Nickel, M., Rosasco, L., and Poggio, T. A. Holographic embeddings of knowledge graphs. In *AAAI*, pp. 1955–1961, 2016.

Perozzi, B., Al-Rfou, R., and Skiena, S. DeepWalk: Online learning of social representations. In *KDD*, pp. 701–710, 2014.

Shi, B. and Weninger, T. ProjE: Embedding projection for knowledge graph completion. In *AAAI*, pp. 1236–1242, 2017.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15(1): 1929–1958, 2014.

Suchanek, F. M., Abiteboul, S., and Senellart, P. PARIS: Probabilistic alignment of relations, instances, and schema. *PVLDB*, 5(3):157–168, 2012.

Sun, Z., Hu, W., and Li, C. Cross-lingual entity alignment via joint attribute-preserving embedding. In *ISWC*, pp. 628–644, 2017.

Sun, Z., Hu, W., Zhang, Q., and Qu, Y. Bootstrapping entity alignment with knowledge graph embedding. In *IJCAI*, pp. 4396–4402, 2018.

Sun, Z., Deng, Z.-H., Nie, J.-Y., and Tang, J. RotatE: Knowledge graph embedding by relational rotation in complex space. In *ICLR*, 2019.

Toutanova, K. and Chen, D. Observed versus latent features for knowledge base and text inference. In *CVSC*, pp. 57–66. ACL, 2015.

Trouillon, T., Welbl, J., Riedel, S., Éric Gaussier, and Bouchard, G. Complex embeddings for simple link prediction. In *ICML*, pp. 2071–2080, 2016.

Wang, Q., Mao, Z., Wang, B., and Guo, L. Knowledge graph embedding: A survey of approaches and applications. *TKDE*, 29(12):2724–2743, 2017.

Wang, Y. and Tian, F. Recurrent residual learning for sequence classification. In *EMNLP*, pp. 938–943, 2016.

Wang, Z., Zhang, J., Feng, J., and Chen, Z. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, pp. 1112–1119. AAAI, 2014.

Wang, Z., Lv, Q., Lan, X., and Zhang, Y. Cross-lingual knowledge graph alignment via graph convolutional networks. In *EMNLP*, pp. 349–357, 2018.

Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.-i., and Jegelka, S. Representation learning on graphs with jumping knowledge networks. In *ICML*, pp. 5449–5458, 2018.

Yang, B., Yih, W., He, X., Gao, J., and Deng, L. Embedding entities and relations for learning and inference in knowledge bases. In *ICLR*, 2015.

Yang, F., Yang, Z., and Cohen, W. W. Differentiable learning of logical rules for knowledge base reasoning. In *NIPS*, pp. 2316–2325, 2017.

Zhu, H., Xie, R., Liu, Z., and Sun, M. Iterative entity alignment via joint knowledge embeddings. In *IJCAI*, pp. 4258–4264, 2017.

Zhuang, Y., Li, G., Zhong, Z., and Feng, J. Hike: A hybrid human-machine method for entity alignment in large-scale knowledge bases. In *CIKM*, pp. 1917–1926, 2017.