Embedding-Based Entity Alignment Using Relation Structural Similarity

Yanhui Peng, Jing Zhang*, Cangqi Zhou, Jian Xu
School of Computer Science and Engineering
Nanjing University of Science and Technology
200 Xiaolingwei Street, Nanjing 210094, China
{118106010712, jzhang, cqzhou, dolphin.xu}@njust.edu.cn

***Abstract*—Entity alignment aims to find entities in different knowledge graphs that semantically represent the same real-world entity. Recently, embedding-based entity alignment methods, which represent knowledge graphs as low-dimensional embeddings and perform entity alignments by measuring the similarity between entity embeddings, have achieved promising results. Most of these methods mainly focus on improving the knowledge graph embedding model or leveraging attributes to obtain more semantic information. However, the structural similarity between the two relations (considering all entities attached on the two relations) in different KGs has not been utilized in the existing methods. In this paper, we propose a novel embedding-based entity alignment method that takes the advantages of relation structural similarity. Specifically, our method first jointly learns the embeddings of two knowledge graphs in a uniform vector space, using the entity pairs regarding to the seed alignments (the alignments already known) that each shares the same embedding. Then, it iteratively computes the structural similarity between the relations in different knowledge graphs according to the seed alignments and the alignments with high reliability generated during training, which makes the embeddings of relations with high similarity closer to each other. Experimental results on five widely used real-world datasets show that the proposed approach significantly outperforms the state-of-the-art embedding-based ones for entity alignment.**

***Index Terms*—entity alignment, knowledge graph embedding, structural similarity**

I. Introduction

Knowledge graphs (KGs), such as WordNet [1], DBpedia [2], and Freebase [3], store knowledge triples in the form of directed graphs, where nodes and edges represent entities and relations respectively. A knowledge triple (*head entity*, *relation*, *tail entity*) (denoted by (h, r, t) in this paper) stands for an edge with two end nodes in the graph, indicating that there exists a specific relationship between the head and tail entities. Recently, the application of KGs in AI domains, including information retrieval [4], recommender systems [5], question-answering [6], [7], and natural language processing [8], has gained more and more attention. There usually exist multiple KGs from overlapped application domains or even within the same domain. KGs are usually incomplete, which sometimes makes it difficult for a single KG to meet the requirements of some AI applications. Integrating heterogeneous knowledge among different KGs via entity alignment can make KGs satisfy wider requirements in different applications. However, this goal is not easy to achieve. Because different KGs may use different languages and symbols to represent entities and relations, it is usually ineffective to align entities using symbolic-based methods [9], [10]. Therefore, embedding-based entity alignment approaches were proposed to solve this problem. In such approaches, entities and relations in KGs are first presented as low-dimensional embeddings, and then entity alignments were performed by measuring the similarity between entity embeddings.

Generally, KG embedding models focus on modeling a single KG to capture the semantics of entities and relations according to the structural information of the KG. As one of the most widely used KG embedding models in entity alignment, TransE [11] interprets a relation as the translation from its head entity to its tail entity. However, such embeddings cannot be directly used to calculate the similarity between entities in different KGs, because they are in different semantic spaces. A common way to solve this problem is to resort to seed alignments, which have already known for the KGs. By making entities with respective to each seed alignment share the same embedding [12], [13], the different KGs can be represented in a unified space. Another train of thought is to use seed alignments to learn a transition from one vector space to another [14]. Therefore, entity alignment heavily relies on existing seed alignments as training data. Unfortunately, seed alignments are usually severely insufficient. To address this issue, some studies, such as BootEA [15], dynamically add entity pairs with high similarity to the training data as seed alignments in the training process. This kind of approach usually needs a mechanism to prevent error propagation that may lead to a sharp decrease in performance. Some methods, such as JAPE [13], use attribute information to assist training, but these methods only perform well when the attributes are heterogeneous and correlations are vague between KGs [15].

We argue that the structural similarity between the relations of different KGs is very helpful for entity alignment, but it has not been leveraged in the existing methods. As we know, the semantics captured by KG embedding is only approximate. For example, TransE makes $\boldsymbol{h} + \boldsymbol{r} = \boldsymbol{t}$ for an observed triple, while what we can actually obtain is $\boldsymbol{h} + \boldsymbol{r} \approx \boldsymbol{t}$, where $\boldsymbol{h}$, $\boldsymbol{r}$, and $\boldsymbol{t} \in R^k$. That is, there are errors in semantics, which adds difficulty to the embedding-based entity alignment. As

The corresponding author of the paper is Dr. Jing Zhang. The source code is available at: https://github.com/pengyanhui/RSimEA.

978-1-7281-8156-1/20/$31.00 ©2020 IEEE
DOI 10.1109/ICBK50248.2020.00027

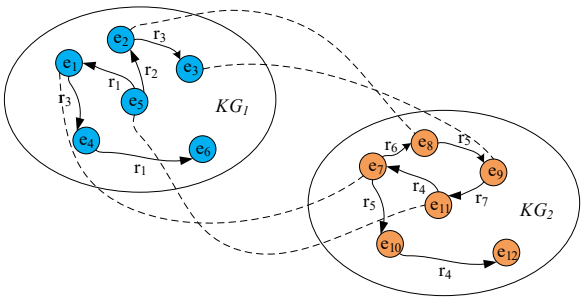


Fig. 1. An illustrative example of entity alignment. KG_1 and KG_2 denote two different knowledge graphs. Entities connected by dashed lines are seed alignments.

shown in Fig. 1, we can know that relation r_1 and relation r_4 are likely to be the same in real world, and their embeddings will be close to each other, as well as r_3 and r_5. With a small error, the entity alignment model may easily find the alignment (e_4, e_{10}). However, as the error accumulates, it may not be so easy to find the alignment (e_6, e_{12}). If we can make the embeddings of relations with high similarity closer to each other, we can alleviate the spreading of accumulated errors. Fortunately, introducing structural similarity between relations into the embedding process can help to archive it. In this study, the structural similarity is defined on the two relevant relations (e.g., r_i and r_4) in different KGs and considers all entities attached to these relations (e.g., e_1, e_4, e_5, e_6 for r_1 and e_7, e_{10}, e_{11}, e_{12} for r_4).

Therefore, we propose a novel embedding-based entity alignment method, namely RSimEA, which utilizes relation structural similarity. Specifically, we jointly learn the embeddings of two KGs in a unified vector space, using the entity pairs regarding to the seed alignments that each shares the same embedding. That is, the two entities of each seed alignment are treated as the same entity in our method. Let r_i be a relation in KG_1, and r_j be a relation in KG_2, $HT_{r_i} = \{(h,t)|(h,r_i,t) \in KG_1\}$ and $HT_{r_j} = \{(h,t)|(h,r_j,t) \in KG_2\}$. We define the structural similarity between r_i and r_j as follows:

$$sim(r_i, r_j) = \frac{2 * \left|HT_{r_i} \cap HT_{r_j}\right|}{|HT_{r_i}| + \left|HT_{r_j}\right|}. \quad (1)$$

According to this definition, if the two relations have many common head-tail entity pairs, they will have a high structural similarity. However, it is obviously not enough to calculate the structural similarity only using seed alignments, because the number of seed alignments is often small. Besides the seed alignments, the entity pairs with high reliability generated during training will also be considered as the same entity, which also can be used in the calculation of the relation similarity. The negative impact of a few wrong entity alignments on the calculation of similarity could be negligible. Thus, the relation similarity will be leveraged for KG embedding.

The main contributions of this paper are summarized as follows:

- We propose that the structural similarity between relations in different KGs is useful for entity alignment. When different KGs are embedded in a unified space, the effect of entity alignment can be improved by making the embeddings of relations with high similarity close to each other.
- We provide a simple but efficient method to measure the structural similarity between relations. For the lack of seed alignments, we use the entity pairs with high similarity in the training process to calculate the relation similarity.
- We conduct extensive experiments to evaluate the proposed approach on the task of entity alignment on five widely used benchmark datasets. The experimental results show that RSimEA has significant improvements compared with the existing state-of-the-art methods.

The remainder of the paper is organized as follows: Section II reviews the related work; Section III introduces some preliminary knowledge; Section IV presents the proposed method; Section V presents the experiments and their analysis; and Section VI concludes the paper.

II. Related Work

A. Knowledge Graph Embedding

Knowledge graph embedding has been demonstrated to be able to capture the semantic information of entities and relations in classical tasks such as knowledge completion and relation extraction [11], [16], [17]. Given a triplet (h, r, t), TransE [11] interprets the relation as a translation r from the head entity h to the tail entity t, making the entity vectors ($\boldsymbol{h}$, $\boldsymbol{t}$) and the relation vector $\boldsymbol{r}$ satisfy $\boldsymbol{h} + \boldsymbol{r} = \boldsymbol{t}$. It is simple and efficient, but it also has some disadvantages such as the inability to represent multi-mapping relations (e.g., one-to-many relations). Some variants of TransE, such as TransH [18] and TransR [19], are proposed to improve TransE on multi-mapping relations by introducing relation-specific entity representation. TransH interprets each relation as a translating operation on a hyperplane, while TransR projects entities from entity space to corresponding relation space and then builds translations between projected entities. TransD [20] considers the diversity of both relations and entities, using two vectors to represent an element (entity or relation). One represents the meaning of the element, and another is used to construct a mapping matrix dynamically. Also, there exist other classic methods for KG embedding [21], [22]. In DistMult [21], a bilinear model, the relation is represented as a diagonal matrix to capture pairwise interactions between the components of h and t along the same dimension. PTransE [23] leverages multiple-step relation paths to help enhance embedding. Among these KG embedding models, TransE becomes the most popular one because of its simplicity, efficiency, and convenience for computing similarity.

B. Knowledge Graph Entity Alignment

Entity Alignment models can be roughly categorized into two groups [24]: string-similarity-based approaches and embedding-based approaches.

Traditional entity alignment methods mainly judge whether entities are the same by measuring string similarity, such as editing distance. LIMES [9] first computes an approximate similarity of each entity pairs by the triangle inequality. Then, if an entity pair has a high approximate similarity, its actual similarity will be computed. The entity pair with the highest actual string similarity is considered as the result entity alignment. Methods only utilizing string similarity are so straightforward that it can not deal with cross-lingual knowledge alignment.

Recently, several embedding-based methods for entity alignment were proposed. For example, MTransE [14] uses TransE to encode entities and relations of KGs with different languages in separated embedding spaces, and then learns transitions for embedding vectors to the cross-lingual counterparts via five alignment models.

IPTransE [25] is an approach that uses PTransE [23] as a basic embedding model for entity alignment, which jointly represents both entities and relations of different KGs into a unified low-dimensional vector space according to a small set of seed entity alignments. As training goes on, it aligns entities according to their semantic distance in the unified semantic space to update embeddings iteratively. However, as it is difficult to guarantee that the newly-aligned entities are correct, error accumulation is a serious problem during iterations. Thus, it needs relation alignments and more seed entity alignments to guarantee the accuracy of embeddings.

JAPE [13] is a joint attribute-preserving embedding approach for cross-lingual entity alignment. It jointly entities and relations of different KGs into a unified vector space and further leverages attribute correlations to refine entity embeddings in the KGs. GCN [26] uses graph convolutional networks to propagate information from neighborhoods to construct entity representations. MuGNN [27] uses a multi-channel graph neural network to learn KG embeddings. MuGNN leverages different channels to reconcile missing relations and exclusive entities, which can enhance entity embeddings, making better use of seed alignments. AlignEA [15] swaps entities of each seed alignments in triples to calibrate the KGs embeddings in a unified embedding space. BootEA [15] dynamically selects entity alignments with high similarity as new seed alignments, and employs an alignment editing method to reduce the influence of error accumulation.

III. Preliminary Knowledge

A. Problem Formulation

A KG can be regarded as a directed graph $G = (E, R, T)$, where E is the set of entities, R is the set of relations, and T denotes the set of all triplets. Each triplet (h, r, t) indicates that there exists a specific relationship between the head and tail entities. Given two heterogeneous KGs $G_1 = (E_1, R_1, T_1)$ and $G_2 = (E_2, R_2, T_2)$, the task of entity alignment aims to find $A = \{(e_1, e_2) \in E_1 \times E_2 | e_1 \leftrightarrow e_2\}$, where $\leftrightarrow$ denotes an equivalence relationship, indicating that e_1 and e_2 are the same real-world entity. A part of the all entity alignments can be obtained manually or by other means (such as simple lexicon-based methods), which we call seed alignments, denoted as $S \in A$. These seed alignments are a bridge connecting the two KGs, which are used as training data in entity alignment task. Let $E_1^{'}$ and $E_2^{'}$ be the unaligned entities that need to be aligned. Entity alignments are usually considered to be one-to-one: an entity in E_1 can be associated with at most one entity in E_2, and vice versa [15].

B. TransE

Like most embedding-based entity alignment models [12], [14], [15], our method is also built on top of the basic KG embedding model TransE [11]. We hereby briefly describe some basic concepts and principles in TransE.

TransE assigns a low-dimensional vector to each element (entity and relation). Given a knowledge triple (h, r, t), the vectors of entities h and t are denoted as $\boldsymbol{h}$ and $\boldsymbol{t}$ respectively, and the vector of r is denoted as $\boldsymbol{r}$, where $\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t} \in R^k$. The semantics of the elements in a KG are contained in the structural information of the graph. In order to preserve the structural information of the graph to capture the semantics of elements, TransE makes $\boldsymbol{h} + \boldsymbol{r} = \boldsymbol{t}$, i.e., $\boldsymbol{h}$ is connected to $\boldsymbol{t}$ by a translation vector $\boldsymbol{r}$. The above idea is realized by a scoring function as follows:

$$f_r(\boldsymbol{h}, \boldsymbol{t}) = \|\boldsymbol{h} + \boldsymbol{r} - \boldsymbol{t}\|_{1/2}, \tag{2}$$

where $\|\boldsymbol{x}\|_{1/2}$ denotes the L1-Norm or L2-Norm of vector $\boldsymbol{x}$. Observed triples will obtain lower scores, while negative triples will obtain higher scores. In order to obtain the embedding that can preserve the structure information of the KG, TransE minimizes the global score of all positive triplets by a margin-based objective function as follows:

$$\mathcal{L}_{\mathcal{G}} = \sum_{(h,r,t) \in T} \sum_{(h',r,t') \in T'} [\gamma + f_r(\boldsymbol{h}, \boldsymbol{t}) - f_r(\boldsymbol{h}^{'}, \boldsymbol{t}^{'})]_+, \tag{3}$$

where we define $[x]_+ = max(x, 0)$, $\gamma > 0$ is a margin hyperparameter, and

$$T^{'} = \{(h^{'}, r, t) | h^{'} \in E\} \cup \{(h, r, t^{'}) | t^{'} \in E\}. \tag{4}$$

T is the set of all observed triples from the training dataset, while $T^{'}$ is the set of corrupted triples, which are used as negative samples. Given a positive sample (h, r, t), the so-called negative sampling replaces the head entity h or tail entity t randomly to generates negative samples $\{(h^{'}, r, t) | h^{'} \in E\}$ and $\{(h, r, t^{'}) | t^{'} \in E\}$. Such a simple method has been proved to be effective in the task of knowledge completion, and it has become one of the most popular embedding models for many specific embedding-based entity alignment methods.

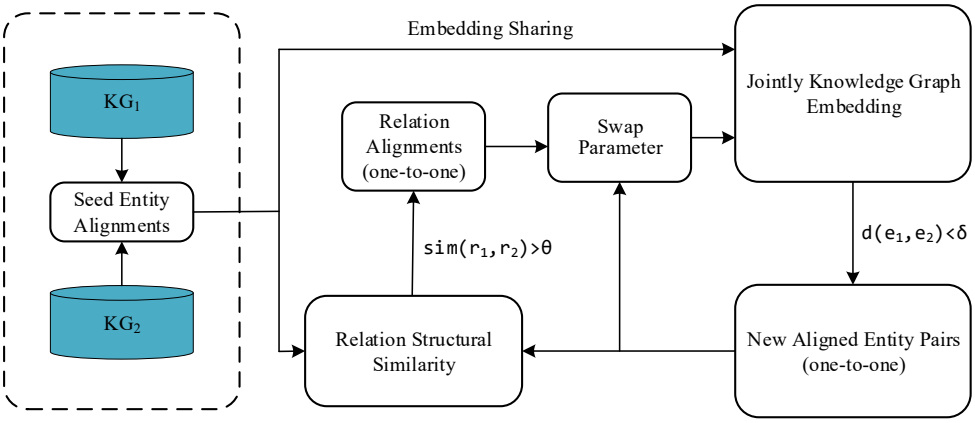


Fig. 2. Overall architecture of our proposed RSimEA.

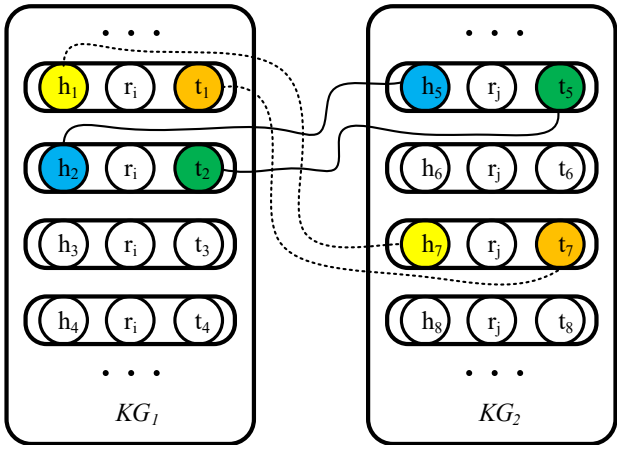


Fig. 3. A simple example of calculating the structural similarity between relations in different KGs.

IV. The Proposed Method

In this section, we will present our proposed RSimEA model in detail. First, we briefly describe the overall architecture of our method. Then, we detail the components of the RSimEA model, including relation structural similarity, embedding learning, and entity alignment.

A. Overall Architecture

The overall architecture of our approach is illustrated in Fig. 2. The dotted rectangle on the left side of the figure is the input data, including two heterogeneous knowledge graphs (KG_1 and KG_2) and some seed alignments. The KG embedding component jointly learns the embeddings of the two KGs in a unified vector space, with the two entities of each seed alignment sharing the same embedding. The seed alignments are not only treated as a bridge to connect the two KGs, but also used to calculate the structural similarity of relations between KG_1 and KG_2. Then, we obtain the likely relation alignments (probably the same relation in the real world) according to the relation structural similarity. The component "Swap Parameter" is a strategy from [15] to leverage alignments for bridging different KGs, which swap aligned entities in their triples to generate new triples. Here, we also use this strategy to swap aligned relations. It can make the embeddings of aligned relations closer to each other. Note that entities of a seed alignment which share the same embedding do not need to be swapped, and aligned entities (relations) will be discarded if they do not satisfy the alignment condition as training goes on.

The training of the model is an iterative process. The similarity between entities is calculated according to their embeddings, and the entity pairs with high similarity will be regarded as new aligned entities, which will be leveraged for KG embedding. But not like seed alignments sharing embeddings, new aligned entities are only used to generate new triples by swapping parameter and compute relation structural similarity, because there is no guarantee that these alignments are correct.

B. Relation Structural Similarity

We have briefly introduced the concept of relation structural similarity in section I, and a more detailed description will be presented here with a simple example.

Fig. 3 shows the triples involving relation r_i and relation r_j of the two heterogeneous KGs. Entities with the same color are the same real-world entity, i.e. seed alignment. ($r_i \in KG_1$ and $r_j \in KG_2$). $HT_{r_i} = \{(h_1, t_1), (h_2, t_2), (h_3, t_3), (h_4, t_4)\}$, $HT_{r_j} = \{(h_5, t_5), (h_6, t_6), (h_7, t_7), (h_8, t_8)\}$. So $HT_{r_i} \cap HT_{r_j} = \{(h_{1/5}, t_{1/5}), (h_{2/7}, t_{2/7})\}$. According to (1), we have $sim(r_i, r_j) = 2 * |HT_{r_i} \cap HT_{r_j}|/(|HT_{r_i}| + |HT_{r_j}|) = 2 * 2/(4 + 4) = 0.5$.

In addition, the new aligned entities are also used to compute the relation structural similarity except for seed

alignments. For example, if the entity alignment component finds out aligned entity pairs (h_3, h_8) and (t_3, t_8) during training, then $sim(r_i, r_j) = 2 * 3/(4 + 4) = 0.75$.

Using relation structural similarity to find relation alignments is not error-prone, because the HT set of a relation is usually large, even if some new entity alignments are wrong, it won't have much impact on the relation similarity.

For the entity alignment module, instead of using the cosine similarity between embeddings as BootEA [15] does, we directly use the Manhattan distance to measure the similarity between entities as follows:

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sum_i |\boldsymbol{x}_i - \boldsymbol{y}_i|. \tag{5}$$

This is because that cosine similarity is defective in entity alignment. Obviously, according to (2), we can know that whether two entities are similar lies in whether the distance between them is small, rather than whether the angle of the corresponding vectors is small. For example, let $\boldsymbol{x} = (1, 2)$ and $\boldsymbol{y} = (2, 4)$, although the cosine similarity between $\boldsymbol{x}$ and $\boldsymbol{y}$ is 1, they are obviously not similar in the vector space.

C. Parameter Swapping

Following [15], we swap aligned entities (relations) in their triples to generate new triples as training data. Such a strategy can make the embeddings of aligned entities (relations) closer to each other in the unified embedding space. Given an aligned entity pair (e_1, e_2) and an aligned relation pair (r_1, r_2), where $e_1 \in E_1$, $e_2 \in E_2$, $r_1 \in R_1$ and $r_2 \in R_2$, we generate the following new triples:

$$\begin{aligned} T_1^+ = &\{(e_2, r, t)|(e_1, r, t) \in T_1\} \cup \{(h, r, e_2)|(h, r, e_1) \in T_1\} \\ &\cup \{(h, r_2, t)|(h, r_1, t) \in T_1\} \end{aligned} \tag{6}$$

$$\begin{aligned} T_2^+ = &\{(e_1, r, t)|(e_2, r, t) \in T_2\} \cup \{(h, r, e_1)|(h, r, e_2) \in T_2\} \\ &\cup \{(h, r_1, t)|(h, r_2, t) \in T_2\} \end{aligned} \tag{7}$$

where T_1 and T_2 denote the positive triple sets of KG_1 and KG_2, respectively. Then, $T_1 \cup T_1^+$ and $T_2 \cup T_2^+$ will be regarded as the positive triple sets of KG_1 and KG_2, respectively. Note that T_1 and T_2 are not changed, and T_1^+ and T_2^+ have nothing to do with the previous ones in each iteration, i.e., aligned entity (relation) pairs are not cumulative.

D. Negative Sampling

A KG only contains positive triplets, unobserved ones are assumed to be negative with large probability, and the way to construct a negative sample is to randomly replace the head or tail entity of an observed triplet, which is called negative sampling. The initial uniform negative sampling method [11] is simple, but many pieces of research [15], [28]–[30] have proved that the negative samples generated by such a method are not good enough. They argue that not all negative samples are equally good, bad ones can make the performance worse. Some negative examples may be obviously wrong, they would contribute little to embedding learning. For example, for the positive triple (*Bill Gates*, *FounderOf*, *Microsoft*), the negative triple (*New York*, *FounderOf*, *Microsoft*) obviously does not hold, while (*Steve Jobs*, *FounderOf*, *Microsoft*) would be a high-quality negative triple.

Reference [29] believes that high-quality negative triplets have large scores (2), and capturing their dynamic distribution is very important. Many negative sampling methods have been proposed according to this idea, among which the self-adversarial negative sampling method [22] dynamically adjusts the weight of negative samples according to their scores as the training goes on. We adopt this negative sampling technique. Specifically, the weight (i.e., probability distribution) of negative triplets for a golden triplet is as follows:

$$p(h_j^{'}, r, t_j^{'}|\{(h_i, r, t_i)\}) = \frac{\exp(\alpha f_r(\boldsymbol{h_j^{'}}, \boldsymbol{t_j^{'}}))}{\sum_i \exp(\alpha f_r(\boldsymbol{h_i^{'}}, \boldsymbol{t_i^{'}}))}, \tag{8}$$

where α is the temperature of sampling. The self-adversarial negative sampling method does not directly obtain negative samples according to this distribution, it still conducts uniform sampling, and then calculates the weight of each negative sample according to (8), which greatly reduces the cost of negative sampling.

E. Optimization

Our method is built on top of the classical KG embedding model TransE [11]. We have briefly described the basic concept and principle of TransE in III-B. But we do not use TransE directly, because the margin-based objective function is defective when it is used for entity alignment. When $f_r(\boldsymbol{h}^{'}, \boldsymbol{t}^{'}) - f_r(\boldsymbol{h}, \boldsymbol{t}) > \gamma$, the corresponding embeddings will not be optimized. It does not guarantee that the positive sample score is very small, i.e., the distance between $\boldsymbol{h}+\boldsymbol{r}$ and $\boldsymbol{t}$ may be long, resulting in larger errors for the semantics.

To solve this problem, we define the logistic loss function for a observed triplet and its negative samples:

$$softplus(\boldsymbol{x}) = \frac{1}{\beta} log(1 + exp(\beta \boldsymbol{x})), \tag{9}$$

$$\begin{aligned} L = &softplus(f_r(\boldsymbol{h}, \boldsymbol{t}) - \gamma) \\ &+ \sum_{i=1}^{n} p(h_i^{'}, r, t_i^{'}) softplus(\gamma - f_r(\boldsymbol{h_i^{'}}, \boldsymbol{t_i^{'}})) \\ &+ \lambda \sum_{e \in E} \|\boldsymbol{e}\|_2 , \end{aligned} \tag{10}$$

where β is a parameter that can adjust the margin between positive and negative sample scores; λ is the regularization coefficient; E is the set of all entities in a KG. Adam [31] is used as the optimizer.

F. Algorithm

The detailed process of the proposed RSimEA method are summarized in Algorithm 1. The input of the algorithm is two KGs to be aligned, the seed alignments S, the maximum training epoch max_epoch, the alignment frequency $align\_freq$, the relation structural similarity threshold θ when aligning relations, and the Manhattan distance threshold δ

Algorithm 1 RSimEA

Input:
Knowledge graph G_1 = (E_1, R_1, T_1);
Knowledge graph G_2 = (E_2, R_2, T_2);
Seed alignments S;
max_epoch, $align\_freq$, θ, δ;
Output:
Entity alignments $A = \{(e_1, e_2) \in E_1 \times E_2 | e_1 \leftrightarrow e_2\}$;

```
Initialize the embeddings randomly;
for all r ∈ R1 ∪ R2 do
    Compute HT_r = {(h,t)|(h,r,t) ∈ T1 ∪ T2};
end for
epoch = 1;
while epoch <= max_epoch do
    Train embeddings one epoch with optimizer Adam [31];
    if epoch%align_freq == 0 then
        Clear A;
        Compute the Manhattan distance d = {d(e_i,e_j)|e_i ∈ E1, e_j ∈ E2};
        Align entities (1-to-1, d(e_i,e_j) < δ) according to d;
        Add aligned entity pairs to A;
        Compute structural similarity s = {sim(r_i,r_j)|r_i ∈ R1, r_j ∈ R2} using S and A by (3);
        Align relations (1-to-1, sim(r_i,r_j) > θ) according to s;
        Swap parameters to generate new triples by (6) and (7);
    end if
end while
return A;
```

when aligning entities. The settings of these parameters in our experiments are presented in Section V.B. The steps of the algorithm can easily mapped into the framework in Fig. 2.

V. Experiments

In this section, we conduct extensive experiments on five widely used benchmark datasets to evaluate the performance of proposed RSimEA method.

A. Datasets

We adopt datasets DBP15K from [13] and datasets DWY100K from [15] in the experiments. The statistical information of these datasets is summarized in Table I.

- DBP15K is built from the multilingual versions of DBpedia, containing three cross-lingual datasets: $\text{DBP}_{\text{ZH-EN}}$(Chinese to English), $\text{DBP}_{\text{JA-EN}}$ (Japanese to English), and $\text{DBP}_{\text{FR-EN}}$ (French to English). Each of the above datasets includes 15,000 entity alignments and some relation alignments.
- DWY100K consists of two large-scale cross-resource datasets extracted from DBpedia, Wikidata and YAGO3: DWY-WD (DBpedia to Wikidata) and DWY-YG (DBpedia to YAGO3). Both the datasets have 100 thousand reference entity alignments, but no relation alignment is provided.

B. Experimental Settings

a) Baselines: To investigate ability of our proposed RSimEA method on entity alignment, we compare the performance of RSimEA and that of seven state-of-the-art models: MTransE [14], IPTransE [25], JAPE [13], GCN [26], MuGNN [27], AlignE [15] and BootEA [15]. The basic principles of these methods are briefly described in II-B. In addition, in order to verify the effectiveness of the proposed relation structural similarity and relation alignment strategy, we remove the corresponding components from RSimEA, and the resulting model is called NonRSE.

TABLE I
Statistical data of DBP15K and DWY100K

Datasets		#Ent.	#Rel.	#Rel tr.	Alignments	
					#Ent.	*#Rel.*
$\text{DBP}_{\text{ZH-EN}}$	ZH	66,469	2,830	153,929	15,000	890
	EN	66,469	2,830	153,929		
$\text{DBP}_{\text{JA-EN}}$	JA	65,744	2,043	164,373	15,000	529
	EN	95,680	2,096	233,319		
$\text{DBP}_{\text{FR-EN}}$	FR	66,858	1,379	192,191	15,000	212
	EN	105,889	2,209	278,590		
DBP-WD	DBP	100,000	330	463,294	100,000	–
	WD	100,000	220	448,774		
DBP-YG	DBP	100,000	302	428,952	100,000	–
	YG	100,000	31	502,563		

b) Evaluation Protocol: By convention, we report three standard evaluation metrics: the Mean of those predicted Reciprocal Ranks (MRR), and the Hits@N (i.e., the percentage of correct alignment ranked at top N, where N = 1, 10). Higher Hits@N and MRR indicate better performance.

c) Hyperparameter Settings: For RSimEA, we set the hyperparameters as follows: temperature of negative sampling $\alpha = 1.0$, fixed margin $\gamma = 6.0$, $\beta = 1.25$ in softplus for both the datasets DBP15K and DWY100K. To make a fair comparison, we set embedding size $k = 75$ following [15]. The batchsize b is set to 1024 for DBP15K, and 2048 for DWY100K. 128 negative triples were sampled for each positive triple. We adopt the learning rate decay strategy. The learning rate is first set to 0.001, it will be set to 0.0001 when the loss value does not decrease as training goes on. And the training spent at most 500 epochs. The entity alignment and relation alignment take place every 10 epochs in the embedding process, i.e., $align\_freq = 10$. We also set two thresholds in Algo. 1 as $\theta = 0.1$ and $\delta = 2.0$.

Following JAPE [13] and BootEA [15], we used 30% of the reference entity alignments as seed alignments and left the remaining as testing data. But we do not use reference relation alignment as training data, even if some relation alignments are provided in DBP15K. That is for fairness, because some previous models did not use relation alignments.

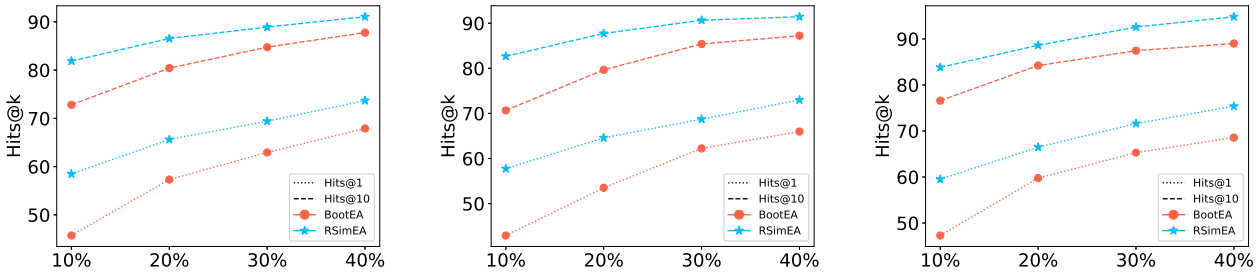
C. Entity Alignment

The results of entity alignment on DBP15K and DWY100K are shown in Table II. We can see that BootEA [15] performs best among the previous models, which benefits from the "bootstrapping" strategy and "parameter swapping" that can automatically correct the wrong alignments. Our NonRSE is comparable to BootEA, which is slightly better in most cases. We believe that this is due to our direct use of Manhattan distance to measure the similarity between entities. MTransE

TABLE II
EXPERIMENTAL RESULTS ON ENTITY ALIGNMENT

Methods	DBP_{ZH-EN}			DBP_{JA-EN}			DBP_{FR-EN}			DBP-WD			DBP-YG		
	H@1	*H@10*	*MRR*	*H@1*	*H@10*	*MRR*	*H@1*	*H@10*	*MRR*	*H@1*	*H@10*	*MRR*	*H@1*	*H@10*	*MRR*
MTransE	30.83	61.41	0.364	27.86	57.45	0.349	24.41	55.55	0.335	28.12	51.95	0.363	25.15	49.29	0.334
IPTransE	40.59	73.47	0.516	36.69	69.26	0.474	33.30	68.54	0.451	34.85	63.84	0.447	29.74	55.76	0.386
JAPE	41.18	74.46	0.490	36.25	68.50	0.476	32.39	66.68	0.430	31.84	58.88	0.411	23.57	48.41	0.320
GCN[a]	41.3	74.4	0.549	39.9	74.5	0.546	37.3	74.5	0.532	50.6	77.2	0.600	59.7	83.8	0.682
MuGNN[a]	49.4	84.4	0.611	50.1	85.7	0.621	49.5	87.0	0.621	61.6	89.7	0.714	74.1	93.7	0.810
AlignE	47.18	79.19	0.581	44.76	78.89	0.563	48.12	82.43	0.599	56.55	82.70	0.655	63.29	84.76	0.707
BootEA	62.94	84.75	0.703	62.23	85.39	0.701	65.30	87.44	0.731	74.79	89.84	0.801	76.10	89.44	0.808
NonRSE	62.19	85.87	0.707	60.65	87.76	0.702	65.53	90.95	0.747	72.01	93.06	0.800	78.58	94.82	0.845
RSimEA	**69.41**	**88.92**	**0.766**	**68.73**	**90.65**	**0.765**	**71.62**	**92.59**	**0.791**	**79.51**	**95.64**	**0.849**	**84.27**	**96.84**	**0.873**

[a] marks the results obtained from [27], other results are taken from [15].



(a) ZH-EN. (b) JA-EN. (c) FR-EN.

Fig. 4. Hits@k on entity alignment w.r.t. proportion of seed alignment.

[14] performs the worst, probably because it embeds two heterogeneous KGs into different vector spaces, respectively and learning the transformation between different embedding spaces is very difficult, especially when the number of the seed alignments is small. IPTransE [25] and JAPE [13] leveraged extra information (relation paths for IPTransE and entity attributes for JAPE), so they achieved better results than MTransE.

GCN is an entity alignment method based on graph convolution network, which achieved relatively good results. MuGNN is a multi-channel graph neural network model, which used rule inference and transfer to get more semantic information, and its performance greatly improved compared with GCN. But these neural network methods are complex and difficult to explain. Large-scale datasets DBP-WD and DBP-YG provide richer semantics. But the performance of MTransE, JAPE and IPTransE on such datasets is worse, indicating that their ability to capture semantics is relatively poor.

RSimEA considerably improved the results of NonRSE after employing relation alignment that uses relation structural similarity, especially on Hits@1 which demonstrated the real precision. In Table II, we can see that Hits@1 on five datasets are improved by 7.22%, 8.08%, 6.09%, 7.05% and 5.69% respectively due to the employ of relation structural similarity. We believe that our RSimEA finds many relation alignments with high accuracy, resulting in greatly reduced error accumulation.

D. Sensitivity to Proportion of Seed Alignments

Whether an entity alignment method is good or not also depends on whether it is sensitive to the number of seed alignments. We know that it is very time-consuming to find seed alignments manually, so a good entity alignment method should use as few seeds as possible to find as many entity alignments as possible.

Following [15], to investigate whether RSimEA is sensitive to the proportion of initial alignments, we gradually increase the proportion of seed alignments from 10% to 40%. Fig. 4 shows the results of our RSimEA and BootEA. We can see that the results became better gradually with the increase of the proportion on all three datasets. In terms of sensitivity to the proportion of seeds, our method performs much better than BootEA. The fewer seeds, the more obvious the advantages of RSimEA. We can see that the Hits@1 of BootEA decreases dramatically as the number of seed alignments decreases, because fewer seed alignments provide less information to align two KGs. Our RSimEA still achieved good results when only using 10% of the total alignments as the seed alignments (Hits@1 on datasets ZH-EN, JA-EN and FR-EN are 58.47%, 57.75% and 59.52% , respectively). This indicates that RSimEA can automatically capture more semantic information of entities and relations than BootEA with limited seed alignments, resulting in the robustness of RSimEA.

VI. Conclusion

In this paper, we proposed a novel embedding-based entity alignment method RSimEA, which employs the structural similarity between relations to improve alignment performance. We provided a simple but efficient technique to calculate relation structural similarity. For the lack of seed alignments, we use the entity pairs with high similarity in the training process to calculate the relation similarity. Extensive experimental results on five widely used benchmark datasets show that the RSimEA model significantly outperforms existing state-of-the-art entity alignment models. A deep investigation into the sensitivity to the proportion of seed alignments further verifies the robustness of RSimEA.

Acknowledgment

This work was supported by the National Key Research and Development Program of China under grant 2018AAA0102002, the National Natural Science Foundation of China under grants 91846104 and 61872186, and the Natural Science Foundation of Jiangsu Province under grant BK20180463.

References

[1] G. A. Miller, "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[2] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer *et al.*, "Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia," *Semantic Web*, vol. 6, no. 2, pp. 167–195, 2015.

[3] K. D. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: A collaboratively created graph database for structuring human knowledge," in *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, 2008.

[4] C. Xiong, R. Power, and J. Callan, "Explicit semantic ranking for academic search via knowledge graph embedding," in *Proceedings of the 26th international conference on world wide web*. International World Wide Web Conferences Steering Committee, 2017, pp. 1271–1279.

[5] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W.-Y. Ma, "Collaborative knowledge base embedding for recommender systems," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 2016, pp. 353–362.

[6] Y. Hao, Y. Zhang, K. Liu, S. He, Z. Liu, H. Wu, and J. Zhao, "An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 221–231.

[7] X. Huang, J. Zhang, D. Li, and P. Li, "Knowledge graph embedding based question answering," in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. ACM, 2019, pp. 105–113.

[8] B. Yang and T. Mitchell, "Leveraging knowledge bases in lstms for improving machine reading," *arXiv preprint arXiv:1902.09091*, 2019.

[12] Y. Hao, Y. Zhang, S. He, K. Liu, and J. Zhao, "A joint embedding method for entity alignment of knowledge bases," in *China Conference on Knowledge Graph and Semantic Computing*. Springer, 2016, pp. 3–14.

[9] A.-C. N. Ngomo and S. Auer, "Limes—a time-efficient approach for large-scale link discovery on the web of data," in *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[10] J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov, "Discovering and maintaining links on the web of data," in *International Semantic Web Conference*. Springer, 2009, pp. 650–665.

[11] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances in neural information processing systems*, 2013, pp. 2787–2795.

[13] Z. Sun, W. Hu, and C. Li, "Cross-lingual entity alignment via joint attribute-preserving embedding," in *International Semantic Web Conference*. Springer, 2017, pp. 628–644.

[14] M. Chen, Y. Tian, M. Yang, and C. Zaniolo, "Multilingual knowledge graph embeddings for cross-lingual knowledge alignment," *arXiv preprint arXiv:1611.03954*, 2016.

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

[16] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *International Conference on Machine Learning*, 2016, pp. 2071–2080.

[17] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2d knowledge graph embeddings," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[18] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *Twenty-Eighth AAAI conference on artificial intelligence*, 2014.

[19] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[20] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, "Knowledge graph embedding via dynamic mapping matrix," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2015, pp. 687–696.

[21] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," *arXiv preprint arXiv:1412.6575*, 2014.

[22] Z. Sun, Z.-H. Deng, J.-Y. Nie, and J. Tang, "Rotate: Knowledge graph embedding by relational rotation in complex space," *arXiv preprint arXiv:1902.10197*, 2019.

[23] Y. Lin, Z. Liu, H. Luan, M. Sun, S. Rao, and S. Liu, "Modeling relation paths for representation learning of knowledge bases," *arXiv preprint arXiv:1506.00379*, 2015.

[24] B. D. Trisedya, J. Qi, and R. Zhang, "Entity alignment between knowledge graphs using attribute embeddings," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 297–304.

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

[26] Z. Wang, Q. Lv, X. Lan, and Y. Zhang, "Cross-lingual knowledge graph alignment via graph convolutional networks," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 349–357.

[27] Y. Cao, Z. Liu, C. Li, J. Li, and T.-S. Chua, "Multi-channel graph neural network for entity alignment," *arXiv preprint arXiv:1908.09898*, 2019.

[28] L. Cai and W. Y. Wang, "Kbgan: Adversarial learning for knowledge graph embeddings," *arXiv preprint arXiv:1711.04071*, 2017.

[29] Y. Zhang, Q. Yao, Y. Shao, and L. Chen, "Nscaching: simple and efficient negative sampling for knowledge graph embedding," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 2019, pp. 614–625.

[30] P. Wang, S. Li, and R. Pan, "Incorporating gan for negative sampling in knowledge representation learning," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.