Cross-Knowledge-Graph Entity Alignment via Relation Prediction

Hongren Huang[a,b], Chen Li[a,b], Xutan Peng[c], Lifang He[d], Shu Guo[e], Hao Peng[a,b], Lihong Wang[e,*] and Jianxin Li[a,b]

[a] *Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University, Beijing, China.*

[b] *The State Key Laboratory of Software Development Environment, Beihang University, Beijing, China*

[c] *The University of Sheffield, Sheffield, United Kingdom*

[d] *Lehigh University, Bethlehem, United States*

[e] *National Computer Network Emergency Response Technical Team Coordination Center of China, Beijing, China*

ARTICLE INFO

Keywords:
Knowledge Alignment
Anchor Relation
Self-training
Data Augmentation
Relation Prediction

ABSTRACT

The entity alignment task aims to align entities corresponding to the same object in different KGs. The recent work focuses on applying knowledge embedding or graph neural networks to obtain entity embedding for entity alignment. However, there are two challenges encountered by these models: one is some models need to design hyper-parameter to balance embedding loss and alignment loss, the other is the limited training data size. In this paper, we propose a novel entity alignment framework named RpAlign (Relation prediction based cross-knowledge-graph entity Alignment) to address these two issues. Specifically, RpAlign transforms the entity alignment task to the KG completion task to solve and does not need to design any extra alignment component. Unlike the existing models that predict aligned entities by using entity vector distance, the RpAlign defines a new relation called 'anchor' for aligned entities, and it predicts new aligned entities based on the relational predictions between the entities. RpAlign employs several data augmentation and improved self-training techniques to mitigate the impact of the data limitation. We conduct experiments on two datasets, and the experimental results show that the RpAlign model significantly outperforms the current state-of-the-art models.

1. Introduction

The large Knowledge Graphs (KGs) such as DBPedia [17], Freebase [3], and Yago [28], have effectively sustained various Natural Language Processing (NLP) applications, e.g., question answering [12], recommendation systems[37], event classification [26], and dialogue generation [35]. These KGs store knowledge as triples (h, r, t), where h is the head entity, t is the tail entity, and r is the relationship between the two entities. Knowledge embedding models are applied to project entities and relations into a low-dimensional vector space so that various applications can directly apply symbolic knowledge in computational form. Recently, cross-domain tasks have attracted rising attention in NLP research, e.g., cross-domain question answering [22], cross-domain language understanding [8], and cross-domain machine reading [9]. The cross-domain tasks require the integration of data information from different data domain sources via the entity alignment method on cross-domain data, such as user alignment [18] for cross-domain social network tasks. So, researchers need to build the cross-domain KGs constructed by the entity alignment approach that can help provide sufficient knowledge assisted information for the cross-domain NLP tasks.

Traditional entity alignment methods manually define various features of entities, which heavily rely on developers' personal experience, leading to high time and labor costs. Recent work has achieved promising performance in various entity alignment tasks by utilizing Knowledge Embedding models and Graph Neural Networks (GNNs). The embedding-based methods employ a triple function to learn the entities' embedding. Among them, classic TransE [4] has been widely used (e.g., MTransE [7]). The GNN-based methods construct an entity network with the relationships between the entities and employ GNNs to learn the entities' embedding. GCN-Align [39] builds a weighted network based on the relationship between entities and employs Graph Convolutional Network (GCN) [15] to learn the entities' embedding. These approaches project multi-source KG entities into a unified semantic vector space. They determine newly aligned entities by calculating the vector distances between entities on two KGs.

However, there are three challenges that existing methods cannot well address. First, the loss functions of existing models (e.g., MTransE [7], BootEA [32], and NAEA [47]), are usually composed of embedding loss and entity alignment loss. Therefore, these models need to manually adjust the weights of the two losses according to the alignment performance, which heavily relies on personal experience and is not suitable for real scenarios. In dealing with different knowledge alignment tasks, it is necessary to design extra hyper-parameters to balance the alignment loss and the knowledge representation learning loss. It seriously restricts such models' generalization performance. Therefore, when this method is directly migrated to different knowledge graph alignment tasks, additional manual experience is needed to adjust the model parameter. It reduces the scope and convenience of these models' practical application. Second, due to the limited scale of labeled data used for training, the performance of most models cannot effectively model the alignment of entities. Because the scale of existing KGs is enormous in reality, it is hard to label

[*] Lihong Wang is corresponding author: wlh@isc.org.cn
ORCID(s): 0000-0003-0179-2364 (L. Wang)

enough aligned knowledge entities due to human resource constraints. It makes the above models unable to learn high-quality knowledge embedding due to limited entity aligned information, and further limits the performance of entity alignment. Third, in reality, the knowledge information in the two knowledge bases is different. The inconsistency of the same entity's knowledge information in different knowledge bases leads to inconsistent neighborhood structure information around the entity. To solve the problem, some embedding-based models like BootEA [32] applied the parameter sharing method on the aligned entity pairs to share all knowledge triples for aligned entities. Some GNN-based method proposes to use graph attention to select and aggregate the neighbor information of nodes to alleviate the challenge. For example, DAEA [29] proposes using a dual attention network, including relational-aware graph attention and hierarchical attention. Hierarchical attention adaptively aggregates low-hierarchy and high-hierarchy information to balance the neighborhood information of the entity. However, the above models only consider sharing and complementing the same entities' knowledge information in different knowledge spaces. They do not share and complement all available knowledge information in two KGs.

In this paper, we propose a novel entity alignment framework named RpAlign (Relation prediction based cross-knowledge-graph entity Alignment), which introduces a specific 'anchor' relation as the relation of alignment entities. First, our model's purpose of introduction of the 'anchor' relation is to merge the two knowledge bases into one knowledge base and use this anchor relation to align the same entity from different source knowledge bases. And RpAlign does not include the extra alignment component, and thus it does not need to specify hyper-parameters that balance multi-source losses. Second, we adopt new data augmentation [27] idea to mine more information for entity alignment. We use pre-aligned entities to mine likely aligned relations between two KGs and utilize them to enrich the size of training triples to help obtain a higher-quality knowledge embedding. So our model can enrich the knowledge information of the two KGs by exchanging the triples from alignment entities and the calculated likely alignment relations. The extra knowledge information can improve the knowledge embedding quality obtained by the model's training. The other purpose for our model's sharing the same relation's knowledge information from different knowledge bases is to make the consistency of the vector space distribution of the two knowledge bases as much as possible. This makes much easier for our model to align and merge two different knowledge bases. Third, the self-training[23] technique is also utilized to modify the composition of the training samples between different epochs to adjust the semantic knowledge space dynamically. The self-training technique can also add the new predicted alignment entity information and enrich the knowledge information by exchanging the new predicted alignment entity's triples in different KGs. Its brought knowledge information can improve the knowledge embedding quality, leading to improved entity alignment task performance.

We evaluated RpAlign on two real-world, large-scale datasets: DBP15K and DWY100K. Experimental results show that our method significantly outperforms the state-of-the-art methods. In terms of Hits@1, RpAlign model is at least 5% better than the existing state-of-the-art methods. Besides, additional visualization and interpretation experiments were conducted to demonstrate the performance of different components in the RpAlign framework. The main contributions of this paper are as follows.

- We propose a novel framework to achieve entity alignment between different KGs through relation prediction on entities. No extra hyper-parameter configuration is required for our framework to balance the losses of various sources in the existing methods. So our model needs to adjust fewer hyper-parameter and avoid performance loss from the failing manual setting of the balance hyper-parameter.
- We propose two new data augmentation technologies: the introduction of 'anchor' relation and parameter sharing operation applied on likely aligned relations. They can supplement the missing knowledge information and make the information of the two KGs consistent. Our model calibrates two KGs with more minor differences by sharing knowledge triples on aligned entities and likely aligned relations.
- The experimental results show that RpAlign significantly outperforms the state-of-the-art models. The experimental results also show that our model can also get promising performance compared with other models when with less labeled training data, and the experimental result decreases much less than that of the current state-of-art models.

The rest of this paper is organized as follows. We discuss the related work about the major existing entity alignment models and knowledge embedding methods in Section2. We describe our RpAlign model in detail in the Section 3. In the Section 4, we report the experimental details and analyze the experimental results. We report our conclusion of this paper's work in the Section 5.

2. Related Work

In this section, we describe the related work that consists of entity alignment and knowledge embedding.

2.1. Entity Alignment

The goal of entity alignment is to align the entities in different KGs. The traditional methods rely on artificial designed features, e.g., [13] employs the OWL properties as features of the entities, [38] utilizes the concept of annotation as entity features. Since human-crafted features are extremely experience-dependent and costly, the performance of such methods is not stable. Besides, some early works applies the ontology information to align entities, e.g., PROMPT [24], and anchor-prompt[25]. Among them, anchor-prompt takes the set of paired term pairs in the two ontologies as input, views the ontology as a graph structure (categories as nodes and slots as edges). It generates a new matching term pair (anchor) by analyzing the path between the aligned entity pairs. However, the early ontology-based methods did not embed the knowledge triple information to align entities. Therefore, they could not use the semantic information of the KG triples, which is essential for the alignment of entities. Recent work proposes embedding-based and GNN-based approaches to learn valuable features automatically. These methods embedded the mixed KG into a unified semantic vector space. They designed a specific entity alignment loss function to minimize vector distance between aligned entities.

Embedding-based approaches use the relationship structure and attributes of the entities to obtain entity embeddings. For example, MTrasnE [7] adopts the TransE [4] to learn the embeddings of two KGs independently and design a mapping function to transform the embeddings between two KGs. MTransE uses the vector distance method to predict the alignment entity. This method's mapping function maps the entity vectors in the source KG vector space to the target KG vector space. The optimization goal of the model is to minimize the vector distance between the aligned entities. JAPE [31] applied the relationship triple and structure information to learn the embedding of two KGs. It also leverages attribute information of entities to help learn more accurate embeddings of entities and applies the vector distance of entities to predict alignment entities. IPTransE [46] is an iterative model that utilizes the parameter sharing method on newly predicted entities to update knowledge embedding. IPTransE model designs the specific 'anchor' relation as the translation bridge of aligned entities in two KGs. The model designs alignment loss from the euclidean vector distance between translated source entities and target entities. It applied PTransE [20] as the KG embedding module to learn embeddings of two KGs. BootEA [32] utilizes self-training also named bootstrapping idea to compute alignment loss by adding new predicted alignment entity pairs to the origin training data. It also applies data augmentation methods on pre-aligned entitiy pairs to genearte cross-KG triples to help solve the problem of limited training data. NAEA [47] employs the attention mechanism to obtain the neighbor-level features, applies the TransE model to get the relation-level features. It finally makes the prediction based on the similarity of the entity features. The OntoEA [41] is the first method that jointly embeds topological structure and knowledge information into a vector representation space for entity alignment. OntoEA uses class hierarchy and class disjointness to prevent false matches. Moverover, some work has applied the human-in-the-loop idea in the task on knowledge graph to reduce the work of the human labeling. For instance, Berrendorf et al. [1] proposed an active-learning based model on entity alignment task. They design labeling strategies from existing heuristics methods and their intuitions to select the most informative examples for the entity alignment model's training.

GNNs have achieved promising performance in practical applications[42, 6, 19]such as recommendation systems, anomaly detection, traffic prediction, and social network. Of course, some works have proposed GNN-based models to solve the task of entity alignment. The GNN-based methods build an entity network based on the relationship structure and acquire the entities' embedding using GNNs (e.g., GCN [15] and GAT [36]). These models first use the knowledge representation learning model and the word vector to obtain the entity's initial vector representation. These methods then apply the relation structure information of two kGs and regard the aligned entities as the bridge connecting two KGs to construct an entity network. They utilize the graph neural network to learn its nodes' embedding. They set the vector distance between the aligned entities in two KGs as the objective optimization function. Among them, the GCN-Align [39] constructs weighted graphs based on the relationship between entities and adopts the GCN to aggregate the entity's neighbor information to get the entities' embedding. Graph Matching [43] first applied monolingual fastText[2] embeddings and the word alignment method proposed in [16] to obtain word representations for entities. It applies the GNN to model the local matching information of entities to derive a graph-level matching vector for entity alignment. MuGNN [5] designs the multi-channel GNNs to encode KGs towards KG completion and pruning exclusive entities and further combines features for obtaining high-quality embedding. The work of AVR-GCN [45] applies the vectorized relational GCN to combine the relationship features and the neighbor information of the entities. The entities' neighborhood structure information in different knowledge bases is usually different, which is a challenge for the GNN-based models to align entities. AliNet [33] introduces distant neighbors to extend the overlapping part of the neighbor structure. The attention mechanism is also used to emphasize helpful distant neighbors and reduce noise. Then It uses the gate mechanism to aggregate the information of direct neighbors and distant neighbors. DAEA [29] proposed a dual attention mechanism that includes relation-aware attention and hierarchical attention. The hierarchical attention mechanism can adaptively aggregate low-hierarchy and high-hierarchy information to balance counterpart entities' neighborhood information and distinguish non-corresponding entities with similar structures. Although the GNN-based models aggregate the neighbor information around entities to aid entity alignment, it does not fully utilize the semantic information of the relations among entities when constructing the network. Simultaneously, in real life, the KG's network scale is massive, the cost of constructing a large-scale network is enormous, so the application scope of these methods is not vast.

Although these methods have achieved promising performance in entity alignment tasks, there are still some shortcomings for the models. Most models need to adjust hyperparameters to balance the entity alignment loss and knowledge embedding loss, which requires a lot of experimental adjustment and mature expert experience. For example, the IPTransE [46] model also defines a special relation connecting the alignment entities from two KGs. However, the IPTransE model still needs to optimize different loss functions to achieve the entity alignment task and the knowledge representation learning task. Our model regards the entity alignment task as judging that two entities satisfy the anchor relations, which means our RpAlign model only need to optimize one loss function. Moreover, they are also affected by limited training data size, which prevents them from improving the quality of knowledge embedding.

BootEA and NAEA also apply the data augmentation method and self-training method [23] to help solve the problem of insufficient data. They just employ these two methods on pre-aligned entities to generate external training data but not on likely aligned relations. They do not fully use the relation structure information in the mixed KG, especially information from likely aligned relations. So it reduces the quality of knowledge embedding of the mixed KG. Besides, these models only consider the difference of aligned entities of two KGs the difference without considering the difference of relations. Embedding-based methods, such as BootEA and JAPE, share the knowledge information among aligned entity pairs; that is, aligned entities share neighbor node information to realize the same neighbor information of entity nodes. GNN-based methods such as AliNet, DAEA design a unique attention mechanism to aggregate neighbor node information at different distances to alleviate entities' inconsistent neighbor structure. Aiming to solve the problem, our model proposed data augmentation technologies to alleviate the challenge caused by two KGs' different relational structure information.

2.2. Knowledge Embedding

Knowledge embedding aims to project entities and relations in the KB into a low-dimensional vector space and used in other downstream tasks. Nowadays, the major existing knowledge embedding methods design a score function for triples in KB, which measures the existence of triples.

Traditional knowledge embedding models, such as TransE [4], TransR [21], TransH[40], and TransD [14], regard triple as geometric relations in vector space, and equip many extra components to enhance the constraints. Unlike these models, some recent work tries to model triples by introducing various functions, e.g., three-way interactions [44], complex space [34], and convolution [10]. In particular, RotatE [30] assumes that the vector of the head entity, relation, and tail entity in any triple (h, r, t) obeys the rule: $\mathbf{h} \circ \mathbf{r} = \mathbf{t}$, so the score function of the model is defined as $f_r(\mathbf{h}, \mathbf{t}) = -\|\mathbf{h} \circ \mathbf{r} - \mathbf{t}\|^2$, where $\circ$ is the Hadamard (element-wise) product operation. RotatE can effectively model three relation patterns: symmetric/antisymmetric, inversion, and composition. RpAlign introduces the 'anchor' relation into the mixed KG and generates many triples with symmetric/antisymmetric patterns, so we choose to leverage the RotatE model's ability to help embed the mixed KG.

3. Methodology

In this section, we first define the problem of entity alignment and then describe RpAlign in detail.

KGs store knowledge as triple (h, r, t), where h and t represents head and tail entities, and r denotes the relation. We formalize a KG as $KG = (E, R, T)$, where E, R, T are the sets of entities, relations and triples. We suppose that $KG_s = (E_s, R_s, T_s)$ and $KG_t = (E_t, R_t, T_t)$ are two KGs to be aligned, and $A = \{(e_{si}, e_{ti})|e_{si} \in E_s, e_{ti} \in E_t\}_{i=1}^{m}$ represents the set of pre-aligned entity pairs, where e_{si} and e_{ti} indicate two prior aligned entities in two KGs, respectively, m is the number of prior aligned entity pairs. The goal of entity alignment task is to predict the remaining unknown aligned entity pairs set $A' = \{(e'_{si}, e'_{ti})|e'_{si} \in E_s, e'_{ti} \in E_t\}_{i=1}^{n}$, where e'_{si} and e'_{ti} indicate two entities to be aligned in two KGs, respectively.

Our model comprises the following modules: data augmentation module, knowledge embedding, self-training learning module. Our data augmentation module mines more knowledge triples from the known alignment entities and relations to help the model learn more accurate knowledge representations. Besides, our model shared the knowledge information of aligned relationships and entities to make the two KG's embedding vector space consistent. Our model uses the knowledge embedding module to learn vector representations of entities and relations and predict new aligned entity pairs. As shown in Figure 1, our proposed RpAlign regards the pre-aligned entities as the bridge connecting two KGs and merge two KGs into one KG. We introduce a new relation r_a named 'anchor' to associate two aligned entities as a new triple into the new mixed KG. Specifically, the triple (h, r_a, t) with the 'anchor' relation means that the head entity h and the tail entity t correspond to the same real-world object. The knowledge embedding module define various score functions to measure the existence of each triple in KG. Hence, we can measure the possibility of alignment for two entities by computing the value of $f_{r_a}(\mathbf{e}_s, \mathbf{e}_t)$. Our model's self-training module continuously discovers and updates the set of aligned entity pairs from the test set. Combined with the data augmentation module to supplement and update the missing knowledge triples, our model can iteratively retrain our model to obtain a more accurate model with better performance.

3.1. Data Augmentation

As shown in Figure 1, we adopt data augmentation technology to produce more cross-KG supervised triples to solve the issue of limited training data. The data augmentation technology of our model can fully exchange the knowledge information of two knowledge bases. So that it can help supplement the missing knowledge information of a single knowledge base and assist the model in learning more accurate knowledge representation. We equip RpAlign with the following data augmentation approaches to generate supervised cross-KG triples for model training.

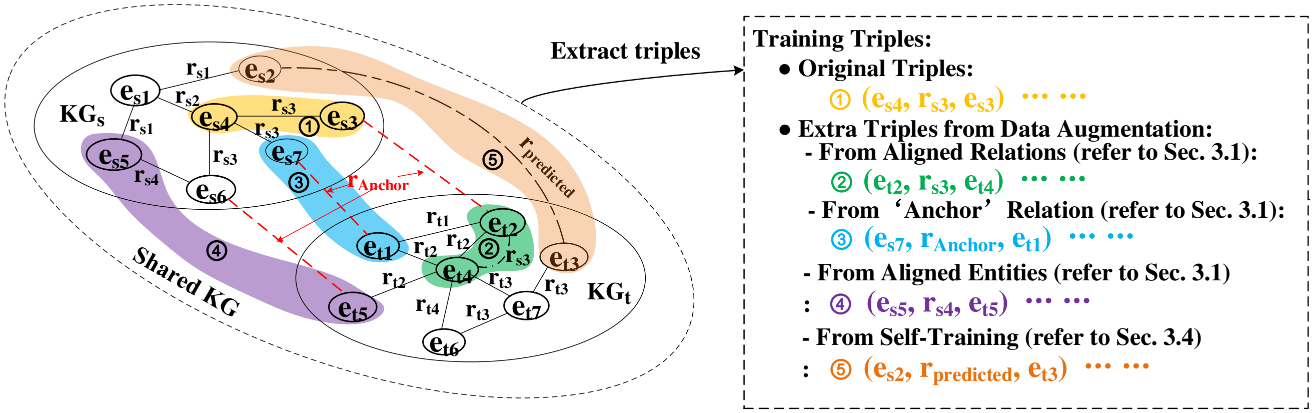

Extract triples
KG$_s$
KG$_t$
Shared KG
e_{s1} e_{s2} e_{s3} e_{s4} e_{s5} e_{s6} e_{s7}
e_{t1} e_{t2} e_{t3} e_{t4} e_{t5} e_{t6} e_{t7}
r_{s1} r_{s2} r_{s3} r_{s4}
r_{t1} r_{t2} r_{t3} r_{t4}
r_{Anchor}
$r_{predicted}$
① ② ③ ④ ⑤
Training Triples:
● Original Triples:
① (e_{s4}, r_{s3}, e_{s3}) ··· ···
● Extra Triples from Data Augmentation:
- From Aligned Relations (refer to Sec. 3.1):
② (e_{t2}, r_{s3}, e_{t4}) ··· ···
- From ‘Anchor’ Relation (refer to Sec. 3.1):
③ (e_{s7}, r_{Anchor}, e_{t1}) ··· ···
- From Aligned Entities (refer to Sec. 3.1)
: ④ (e_{s5}, r_{s4}, e_{t5}) ··· ···
- From Self-Training (refer to Sec. 3.4)
: ⑤ (e_{s2}, $r_{predicted}$, e_{t3}) ··· ···

Figure 1: The visualization of SARP framework and how three data augmentation methods combined with self-training technology produce new triples. KG_s and KG_t are two different KGs. The red dashed lines indictae the ‘anchor’ relation.

3.1.1. Triples From Likely Aligned Relations

Because the triple knowledge information of the same relation in different entities is different, the semantic information learned by the same relation in different KG vector spaces is also different. Therefore, it is difficult to align the same entities in two KGs with different semantics for the same relations. In addition, the lack of knowledge information reduces the quality of the representation vector of the entire KG. To solve the problem, our model adopts the sharing operation on the same relations to make two KGs carry the same knowledge information as possible and discover more missing knowledge triples.

We first infer the likely aligned relations on two KGs that may correspond to the same relationship in the real world. These relations assist our model to exchange triple information related with these alignment relations between the two KGs. On the one hand, alignment relations have identical triples, which can achieve the goal of learning knowledge vectors with the same semantics for aligned relations. Furthermore, the similarity of vectors of aligned relations can help the model calibrate two KGs’ vector space.

RpAlign then calculates the probability of how a relation in one KG can be replaced by another relation in a different KG to obtain its possible aligned relations. The probability that the relation r_s in KG_s can be replaced by relation r_t in KG_t depends on the number of triples they share. We count the number of triples they share as follows:

$$\begin{aligned} c(r_s, r_t) = &count(\{(h_s, r_s, t_s) | (h_s, r_s, t_s) \in T_s, \\ &(h_t, r_t, t_t) \in T_t, (h_s, h_t) \in A, (t_s, t_t) \in A\}) \end{aligned} \quad (1)$$

where A is the set of pre-aligned entity pairs. For a relation pair (r_s, r_t), where $r_s \in R_s$ and $r_t \in R_t$, we calculate the probability that relation r_t is the replacement of relation r_s as follows:

$$p_{ar}(r_t|r_s) = \frac{c(r_s, r_t)}{\sum_{r_i \in R_t} c(r_s, r_i)}. \quad (2)$$

RpAlign brings about the set of aligned relation pairs for the relations in KG_s:

$$R_{A_s} = \{(r_s, r_t) | r_s \in R_s, r_t = \max_{r_i \in R_t} p_{ar}(r_i|r_s)\}, \quad (3)$$

where R_s and R_t are sets of the relations of KG_s and KG_t. We also calculte $p_{ar}(r_s|r_t)$ which represents the probability that r_s is the replacement of r_t, and generate the aligned relation pairs R_{A_t} for KG_t. Because different languages have inconsistent definitions of the same relationship, some relationships in the source knowledge base may correspond to multiple relationships in the target knowledge base. To avoid the above problem introducing too much noise, our model finds a relationship with the highest alignment probability as an alignment relationship in the target KG for each relationship in the source KG.

We can further replace the relations with aligned relations in their relevant triples to generate supervised cross-KG triples. Given each pair of aligned relations $(r_s, r_t) \in R_{A_s}$, we replace relation r_s with relation r_t in its relevant triples to produce the following supervised triples:

$$T_r^s = \{(h, r_t, t) | (r_s, r_t) \in R_{A_s}, (h, r_s, t) \in T_s\}. \quad (4)$$

We can also obtain more supervised cross-KG triples T_r^t from likely aligned relations on KG_t. RpAlign adds these supervised cross-KG triples set T_r^s and T_r^t to the two KGs’ original training triples.

The information of these triples can help supplement a single knowledge base’s missing information. Some related entities can obtain more knowledge information directly so that our model can learn more accurate entity embedding for entity alignment tasks. The main work of these triples is make sure the same knowledge information for likely ‘aligned’ relations in two KGs, which assist in calibrating embedding of two KGs by relations alignment.

3.1.2. Triples From ‘Anchor’ Relations

We introduce the ‘anchor’ relation r_a as the relation for aligned entities on two KGs. So we can generate the following supervised triples that can be added to the original training triples:

$$T_a = \{(e_s, r_a, e_t) \cup (e_t, r_a, e_s) | (e_s, e_t) \in A\}, \qquad (5)$$

where A is the set of aligned entity pairs, E_s and E_t are set of entities of two KGs. Hence, the number of triples with 'anchor' relation in our training data is larger than that of all entities on two KGs, which alleviates the problem of the limited pre-aligned entity pairs. These triples can make the model learn the score function for the anchor relation. The model can apply this score function of the relation to measure the existence of aligned entities. Thus, RpAlign would produce new triples with the three relation patterns: symmetric/antisymmetric, inversion, and composition. The triples generated by the anchor relation have multiple relation patterns, which brings challenges for our model to learn the scoring function of the relation.

3.1.3. *Triples From Aligned Entities*

Because the triple knowledge information of the same entity in two KGs is different, the semantic information learned by the same entity in different KG vector spaces is also different. Therefore, it is difficult to align these entities. In addition, the lack of knowledge information reduces the quality of the representation vector of the entire knowledge base. So our model adopts the sharing operation on the same entity in two KGs to help calibrate the representation vector spaces of two KGs.

Similar to BootEA [32], we also apply parameter sharing [11] operation on aligned entities by exchanging them in their relevant triples to generate more supervised triples. The operation can keep the semantic consistency of aligned entities in the unified knowledge embedding vector space. Given a pair of aligned entities $(e_s, e_t) \in A$, we can produce the new following triples:

$$\begin{aligned} T_{sw} = &\{(e_t, r, t) | (e_s, r, t) \in T_t\} \cup \{(h, r, e_t) | (h, r, e_s) \in T_s\} \\ &\cup \{(e_s, r, t) | (e_t, r, t) \in T_s\} \cup \{(h, r, e_s) | (h, r, e_s) \in T_t\}, \end{aligned}$$

where T_s and T_t are triples of KG_s and KG_t, respectively. The triples T_{sw} can be added to origin training triples to calibrate embeddings of entities on two KGs. These triples are added to the model to ensure that aligned entities have the same relational structure information in the merged KG.

3.2. Learning Knowledge Embedding

After the RpAlign framework obtains the external triples generated by the data augmentation module, RpAlign uses the knowledge embedding module to learn embeddings of entities and relations in the mixed KG. According to section 3.1.2, RpAlign introduces the relation r_a as the relation between two aligned entities on the mixed KG, which generates the external triples with symmetric/antisymmetric relation pattern. Because pRotatE can more effectively model symmetric/antisymmetric relation patterns than other existing knowledge embedding models. RpAlign chooses the pRotatE as its knowledge embedding module. pRotatE is a variant of RotatE model, modulus of the entity embeddings are constrained: $|\mathbf{h}_i| = |\mathbf{t}_i| = C$, where $\mathbf{h}_i$ and $\mathbf{t}_i$ denote head and tail entities, and the symbol $|\cdot|$ denotes the cardinality of a vector. It define the distance function for each triple (h, r, t): $d_r(\mathbf{h}, \mathbf{t}) = \|\mathbf{h} \circ \mathbf{r} - \mathbf{t}\|^2$. The optimization objective is to minimize the distance function values of all triples. The knowledge embedding module of RpAlign adopts the negative sampling technique to optimize the model effectively. The final loss function of the knowledge embedding module for RpAlign is:

$$\begin{aligned} L = &- \log \sigma(\gamma - d_r(\mathbf{h}, \mathbf{t})) \\ &- \sum_{i=1}^{n} p(h'_i, t'_i | h, r, t) \log \sigma(d_r(\mathbf{h}'_i, \mathbf{t}'_i) - \gamma), \end{aligned} \qquad (6)$$

where σ is the sigmoid function, γ is a fixed margin, and $p(h'_i, t'_i | h, r, t)$ is the distribution that knowledge embedding module sample negative triples.

3.3. Predicting New Aligned Entities

After obtaining knowledge embedding of the mixed KG, RpAlign uses embedding of entities and 'anchor' relation r_a to predict the relations between entities and then applies the relation prediction results to find aligned entities. For each unaligned entity e_s in KG_s, RpAlign adopts the following formula to predict the probability distribution of potentially corresponding alignment entity in KG_t:

$$P((e_s, e_t) \in A' | e_s \in E_s, e_t \in E_t) = \frac{\exp f_{r_a}(\mathbf{e}_s, \mathbf{e}_t)}{\sum_{e_i \in E_t} \exp f_{r_a}(\mathbf{e}_i, \mathbf{e}_t)}, \qquad (7)$$

where E_t is the set of entities in KG_t and r_a is the 'anchor' relation between unaligned entities.

3.4. Improved Self-Training

As shown in Figure 1, we also adopt the self-training paradigm to help solve the issue of limited training data. The concept of self-training is to label the unlabeled data from the prediction results after each round of training, and then add new labeled data to the origin training data to retrain the model at the next training iteration. We perform one-to-one entity alignment labeling on unaligned entities in two KGs. For each unaligned entity e_s in KG_s, we find its most likely aligned entity e_t from KG_t by the value of $f_{r_a}(\mathbf{e}_s, \mathbf{e}_t)$. Then RpAlign can produce the following new aligned entity pairs:

$$A_{new} = \{(e_s, e_t) | e_s \in E_s, e_t = \max_{e_t \in E_t} f_{r_a}(\mathbf{e}_s, \mathbf{e}_t)\}. \qquad (8)$$

Note that conflicts may happen during the labeling process, such as (e_1, e_t) and (e_2, e_t) both exist in A_{new}, which also means there is a wrong entity alignment pair. Hence, the conflicts will introduce error training data, which in turn will reduce the effectiveness of RpAlign. So we would like to choose the one with a higher probability of presence to solve the above conflicts. Formally, we compute the following probability difference to solve the conflict:

$$\Delta(e_1, e_2, e_t) = f_{r_a}(\mathbf{e}_1, \mathbf{e}_t) - f_{r_a}(\mathbf{e}_2, \mathbf{e}_t). \qquad (9)$$

$\Delta(e_1, e_2, e_t) > 0$ means that the existence probability of aligned entity pair (e_1, e_t) is larger than (e_2, e_t), which indicates that we should choose (e_1, e_t) to add to A_{new}. Besides, after each round of training, the self-training learning module will update the aligned entities in the target KG corresponding to each source KG's entities in the test set. If the newly predicted alignment entity pair's scoring function value is higher than that of the old alignment entity pair for an entity in the source KG, the target alignment entity corresponding to this entity will be updated.

We sort the newly obtained aligned entity pairs in A_{new} by using the value of $f_{r_a}(\mathbf{e}_s, \mathbf{e}_t)$ for each entity pair (e_s, e_t), and we preserve the top confident $N\%$ of aligned entity pairs in A_{new} and drop the others. N is the hyper-parameter our model sets to avoid adding much noise at the early self-training phase. Before self-training starts, the KG's embedding learned by the model contains insufficient knowledge information, so the newly predicted aligned entity pair set contains more errors. Therefore, in the early stage of the self-training module, N will be set to be minor, and those new predicted alignment entity pairs with a higher possibility are added to the known alignment entity pair set. After our model then learns more accurate KG embedding, N is set larger to generate more triples to improve the performance.

Unlike BootEA [32], we do not directly use the new obtained aligned entity pairs to calculate alignment loss for optimal embedding. The purpose of designing the self-training module of our model is to use the newly acquired aligned entity pairs to dig out more unknown knowledge information and then generate more knowledge triples. So, RpAlign combines the self-training method with data augmentation methods as described in Section 3.1, that adds newly computed aligned entity pairs to original ones to generate more triples for the next training iteration so that RpAlign can iteratively improve the quality of knowledge embedding. These generated triples are add to help supplement some missing knowledge information that a single KG may miss, so the model can learn knowledge embedding with much more knowledge information. These triples are also utilized to calibrate aligned entities' embedding by making them share the same relational structure information. Moreover, since the relationship between the two knowledge bases is not one-to-one, when the number of alignment entities is small, there is a certain chance that the triples generated by the alignment relations in the data augmentation module may be wrong. The self-training learning mechanism can continuously generate new shared alignment entity pairs, improving the data augmentation modules' accuracy of the new generated triples. It means that the self-training module can reduce the noise from the data augmentation module. In summary, our RpAlign's self-training module can help the data augmentation module generate more cross-KG triples. It can also alleviate the noise problem caused by the few labeled data and improves the data's quality generated by the data augmentation module.

4. Experiments

We used PyTorch as a deep learning framework to develop RpAlign and conducted experiments on a computer with a Tesla V100-PCIE-32GB GPU and 256 GB memory.

4.1. Datasets

In order to evaluate our proposed RpAlign and its variant model on various entity alignment tasks, we use the following two datasets DBP15K and DWY100K in the experiments: **DBP15K** [31] is built by using multilingual versions of the KG named DBPedia. DBP15K includes three datasets: DBP$_{ZH-EN}$ (Chinese-English), DBP$_{JA-EN}$ (Japanese-English), and DBP$_{FR-EN}$ (French-English). Each dataset contains 15 thousand aligned entity pairs, which are reference alignment links from entities of the English version to the other three versions. **DWY100K** [32] is created from three KGs: DBPedia, YAGO, and Wikidata. It includes two datasets DBP-WD, DBP-YG. The DBP-WD contains 100 thousand aligned entity pairs that are extracted from reference alignment links from DBPedia to Wiki. The DBP-YG includes 100 thousand aligned entities pairs from reference alignment links between entities of DBPedia and YAGO. Table 1 summarizes the statistics of the experiments datasets.

4.2. Implementation and Parameter Settings

For all datasets, we set the dimension of knowledge embeddings as 500. We sample 32 negative triples for each positive triple and set γ as 24 and α as 1.0 for the knowledge embedding module of RpAlign. We set the number of training steps in each training iteration for DBP15K as 250,000, and 100,000 for DWY100K. The training batch size in each step for DBP15K and DWY100K is 128 and 1024, respectively. The training iteration number is 100 for two datasets. We configure the learning rate as 1×10^{-4} during the first two training iterations, 2×10^{-5} range from 3rd to 20th training iteration, 4×10^{-6} from 21st to 100th iteration.

To leverage the self-training paradigm in our training process. We choose top confident $N\%$ of the new aligned entity pairs predicted from the previous training iteration to retrain our model. In particular, N is set to 0 during the first three training iterations. From the 4th to the 12th training phase, N is set as $\{20, 30, 40, 50, 60, 70, 80, 90, 100\}$, respectively, and from 13th to 100th training iteration, N is 100. N's value is set from small to large during the training iterations because the knowledge embedding is not good enough at the beginning of the training process. So that RpAlign can avoid introducing too much noise to the original training data.

Following JAPE [31], we use Hits@1, Hits@10, Mean Reciprocal Rank (MRR) as metrics for comparison. Hits@k is the proportion of correct answers ranked in top-K, especially Hits@1 can be viewed as accuracy metric. MRR is the average of the reciprocal ranks of results. The higher values of the three metrics indicate better entity alignment performance. As in the work of JAPE, we selected 30% reference alignment links as the training data.

Table 1
Statistics of the datasets.

DataSet	DBP$_{ZH-EN}$		DBP$_{JA-EN}$		DBP$_{FR-EN}$		DBP-WD		DBP-YG	
	Chinese	English	Japanese	English	French	English	DBpedia	Wikidata	DBpedia	YAGO
#Ent	66,469	98,125	65,744	95,680	66,858	105,889	100,000	100,000	100,000	100,000
#Rel	2,830	2,317	2,043	2,096	1,379	2,209	330	220	302	31
#Attr	8,113	7,173	5,882	6,066	4,547	6,422	351	729	334	23
#Rel.tri	153,929	237,674	164,373	233,319	192,191	278,590	463,294	448,774	428,952	451646
#Att.tri	379,684	56,755	354,619	497,230	528,665	576,543	381,166	789,815	451,646	118,376

For a ablation study to discover the impact about data augmentation mthods that our model applied, we first designed two variants of RpAlign, named **wo/align-relation**, and **wo/align-entity**. **wo/align-relation** is an implementation of RpAlign that learns knowledge embeddings without using externally generated triples from likely aligned relations described in 3.1.1. **wo/align-entity** is an implementation of RpAlign that learns knowledge embeddings without using externally generated triples from aligned entities described in 3.1.3. To discover the impact of self-training described in 3.4, we designed variants named **wo/self-training** that do not applied with the self-training mechanism. For a ablation study to explore the impact of knowledge embedding module, we also use the TransE, Distmult, and ComplEx as the knowledge embedding module of RpAlign named **repl.TransE**, **repl.Distmult** and **repl.ComplEx**. Moreover, for a ablation study to discover the impact of introduction of 'anchor' relation described in 3.1.2, we designed variants named **repl.cos**. **repl.cos** predicts new aligned entities by using the cosine similarity between entities instead of using relation prediction results between entities described in 3.1.2. We ran RpAlign and its variants five times and reported the average as the results.

4.3. Baseline Methods

To evaluate our framework, we choose the current five state-of-art methods as baselines, where the differences have been discussed in Section 2. The functionality of the main modules and their implementation details are following:

- **MTransE** [7] separately projects two KGs into two vector spaces by TransE, and designs a map function for translating entities and relations between two KGs.

- **IPTransE** [46] adds relational path information to the knowledge embedding model. It is an iterative approach that also applies the parameter sharing method on the newly predicted aligned entities.

- **JAPE** [31] combine two KGs to make a mix one by the aligned entity pairs. It applies relational structure information and attribute information to obtain a unified semantic embedding of the mixed KG for entity alignment.

- **BootEA** [32] adopts the bootstrapping idea to iteratively labels likely aligned entity pairs to alleviate the limited training data, and reuse these new predicted aligned entity pairs to update alignment loss for more accurate knowledge embedding.

- **NAEA** [47] utilizes neighborhood subgraph-level information of entities and uses attention mechanism to get neighborhood-level features of entities. It adopts the relational structure features and neighborhood-level features to get a unified semantic embedding for entity alignment.

- **AliNet** [33] introduces distant neighbors to expand the overlap between its neighbor structures, and it uses an attention mechanism to highlight useful distant neighbors and reduce noise. And it proposes a relationship loss function to learn the embedding of entities for the entity alignment task.

- **DAEA** [29] proposed the relation-aware graph attention and hierarchical attention. The relation-aware graph attention selectively aggregate and form multi-hierarchy neighborhoods. The hierarchical attention adaptively aggregates low and high hierarchy information. It helps balance the neighborhood information of corresponding entities and distinguish non-corresponding entities with similar structures.

We directly refer to the results of baselines from their papers, because we used the same datasets and we set the same proportion of seed alignment entity in experiments.

4.4. Result Analysis

The results of entity alignment on DBP15K and DWY100K are listed in Table 2. We observe that the RpAlign outperforms all baselines on all five datasets. Specifically, RpAlign significantly improves performance on Hits@1. Compared to the best baseline NAEA [47], the RpAlign improves Hits@1 by at least 8% on the three datasets included by DBP15K and by 6% on the two datasets belonging to DWY100K. Similarly, RpAlign could also get higher performance than the baseline methods on the two metrics, Hits@10 and MRR.

Besides, we observe that MTransE [7] gets the worst entity alignment performance because MTrasnE separately projects two KGs into two vector spaces, and it leads to information loss when translating entities. Besides, MTRansE ignores the knowledge differences of entities and relationships in different knowledge bases compared with other models. Moreover, its learned entity vector distribution of the embedding representation space of the two learned knowledge

Table 2
Comparison results on DBP15K and DWY100K.

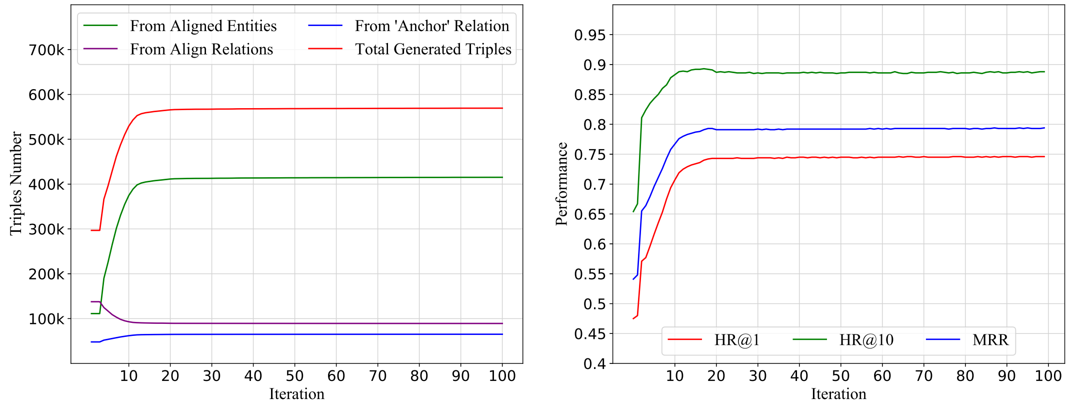
Model	DBP_{ZH-EN}			DBP_{JA-EN}			DBP_{FR-EN}			DBP-WD			DBP-YG		
	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR
MTransE	30.83	61.47	0.364	27.86	57.45	0.349	24.41	55.55	0.335	28.12	51.95	0.363	25.15	49.29	0.334
IPTranse	40.59	73.47	0.516	36.09	69.26	0.474	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
BootEA	62.94	84.75	0.703	62.23	85.39	0.701	65.30	87.44	0.731	74.79	89.94	0.801	76.10	89.44	0.808
NAEA	65.01	86.73	0.720	64.14	87.27	0.718	67.32	89.43	0.752	76.70	91.79	0.817	77.86	91.25	0.821
AliNet	53.90	82.60	0.628	54.90	83.10	0.645	55.20	85.20	0.657	69.00	90.80	0.766	78.60	94.30	0.841
DAEA	56.76	88.30	0.677	57.59	89.23	0.683	58.04	91.16	0.695	-	-	-	-	-	-
RpAlign	**74.78**	**88.86**	**0.794**	**72.96**	**89.02**	**0.782**	**75.22**	**89.96**	**0.801**	**82.64**	**93.36**	**0.862**	**83.84**	94.51	**0.872**
Gain	9.77	2.13	0.074	8.82	1.75	0.064	7.90	0.53	0.049	5.94	1.57	0.045	5.98	3.26	0.051

Table 3
Results of ablation experiment.

	DBP(zh)			DBP(ja)			DBP(fr)			DBP-WD			DBP-YG		
	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR
RpAlign	74.78	88.86	0.794	72.96	89.02	0.782	75.22	89.96	0.801	82.64	93.36	0.862	83.84	94.51	0.872
wo/align-relation	63.19	81.38	0.690	57.40	78.27	0.639	59.71	79.41	0.659	71.75	87.17	0.767	77.39	89.84	0.812
wo/align-entity	68.35	85.93	0.742	66.08	84.83	0.722	66.52	85.31	0.727	73.41	87.08	0.778	78.62	91.53	0.828
wo/self-training	54.74	80.09	0.636	54.60	80.13	0.634	55.05	83.00	0.646	65.61	85.30	0.724	68.26	88.81	0.753
repl.cos	64.29	81.57	0.697	69.02	86.31	0.748	70.29	88.24	0.763	79.28	92.40	0.837	81.60	94.84	0.816
repl.TransE	70.88	87.09	0.762	63.24	87.53	0.714	64.46	89.03	0.644	78.69	91.57	0.786	74.60	92.97	0.807
repl.Distmult	16.69	23.53	0.190	18.30	22.84	0.200	14.29	19.34	0.161	63.26	78.07	0.685	40.60	49.87	0.406
repl.ComplEx	20.06	28.90	0.229	27.88	31.30	0.291	18.58	23.70	0.205	62.30	75.94	0.668	36.28	52.07	0.413

bases is quite different. Therefore, it is difficult for the model to learn a good translation function that can accurately transform all entity vector spaces. Besides, few training triples is available, and the learned knowledge vector is not accurate enough, which also affects the model's performance.

IPTransE [46] and JAPE [31] get better performance because they applies additional information for knowledge embedding. IPTransE adds additional path information to learn embedding of entities and performs parameter sharing operations on the entity to solve the difference of aligned entities on two KGs. JAPE adds the attribute information of entity to learn the embedding of entities for knowledge alignment. The entity alignment model is a challenge that difference of knowledge information of entities in different KGs. GNN-based models such as AliNet [33] and DAEA [29] use graph attention mechanisms to aggregate the entity's neighbor information to solve the problem. For example, DAEA designs a dual attention mechanism in which hierarchical attention adaptively aggregates low-hierarchy and high-hierarchy information to balance the corresponding entity's neighborhood information. In addition to using parameter sharing on entities to realize information sharing to solve the difference of entities on two KGs, BootEA and NAEA also use a self-training learning mechanism to alleviate insufficient training data. For example, BootEA [32] adopts the bootstrapping idea, and it iteratively adds new likely aligned entities to compute entity alignment loss to help calibrate the knowledge embedding of two KGs. So they outperform AliNet and DAEA. NAEA [47] could achieve the highest results except for RpAlign. Because it combines relation-level and neighborhood-level information to obtain a better knowledge embedding.



(a) Triples number with iteration

(b) Performance with iteration

Figure 2: Visualization of impacts of data augmentations and sensitivity study on DBP_{ZH-EN}[1]. (a) is the the changes in numbers of generated triples. (b) is the changes of performance.

The reasons why our model outperforms the baseline models are as follows: First, the data augmentation module and the self-training learning module of our model supplement external knowledge information for two KGs as much as possible. It can alleviate the problem of insufficient labeled training data. Second, our data augmentation technology also makes the information of the two knowledge spaces consistent. Unlike other models only solve the differences in the alignment entities in different knowledge bases. Our model also considers the differences in the alignment relations in two KGs. Our model's obtained entity embedding in the two knowledge bases are more consistent and easier to align. Third, different loss functions bring different learning goals for the model's embedding parameter learning, and the baselines must balance the two loss functions that would train the model with different purposes. Our model only has a knowledge representation loss function to learn, which avoids performance loss caused by the incorrect setting of hyper-parameters that balance different losses.

4.5. Ablation Study

4.5.1. Impacts of Data Augmentation

To explore the importance of the data augmentation module of RpAlign, we conducted the following ablation experiment. We removed the three data augmentation module described in 3.1.1, 3.1.2 and 3.1.3 from RpAlign. The results of the experiment are shown in Table 3, and we have following observations from it:

First, We can observe that RpAlign can obtain an average of 18.98% on Hits@1 improvement compared to wo/align-relation, and average 15.75% higher on MRR. The reason is that the cross-KG triples generated by parameter sharing on likely aligned relations can help calibrate embedding of two KGs. First, the sharing operation on calculated alignment relationship can make consistent for vector distribution of the relations between two KGs. It can help calibrate the embedding of relations on two KGs, which then can help calibrate the embedding of entities. Second, these alignment relation sharing operations can supplement missing knowledge information for a single KG so that our model can get more accurate knowledge embedding. Taking the DBP$_{ZH-EN}$ entity alignment task as an example, the number of knowledge triples that these relationships bring is about $89k$, and the ratio of the original triples in the source KG is about 0.57. This means that these alignment relations can greatly help calibrate and overlap of two knowledge embedding spaces.

Second, RpAlign can achieve about 7.20% on Hits@1 improvement compared with repl.cos, and average 6.65% on MRR. If the cosine similarity is used to find the source entity's alignment target entity, there are some noise entities with a similar cosine distance to the target alignment entity. These noise entities increase the model's difficulty identifying the correct target alignment entity from the source entity. However, our model learns a special anchor relationship to describe entity alignment, i.e., modeling the 'anchor' relationship by introducing learnable triplet scoring functions. This score function, driven by alignment loss, can establish nonlinear mappings for the aligned entities, which is more robust and has stronger generalization performance. Therefore, the alignment operation of our model can consider more complicated alignment entity pair.

Third, RpAlign can achieve 10.42% on Hits@1 improvement compared to wo/align-entity, average 8.32% higher on MRR. This is because RpAlign model generates new triples to ensure that the aligned entities have exactly the same structural information, and these triples complement the missing knowledge for a single knowledge base. Taking the DBP$_{ZH-EN}$ entity alignment task as an example, at the 20th training iteration, the number of generated triples from aligned entities is about $400k$. It is more than two times over origin triples that two KGs originally contained.

4.5.2. Impacts of Improved Self-Training

To better understand the effects of our improved self-training method, we conduct the ablation experiment and show it on the table 3. We can have the following observations from the comparison results:

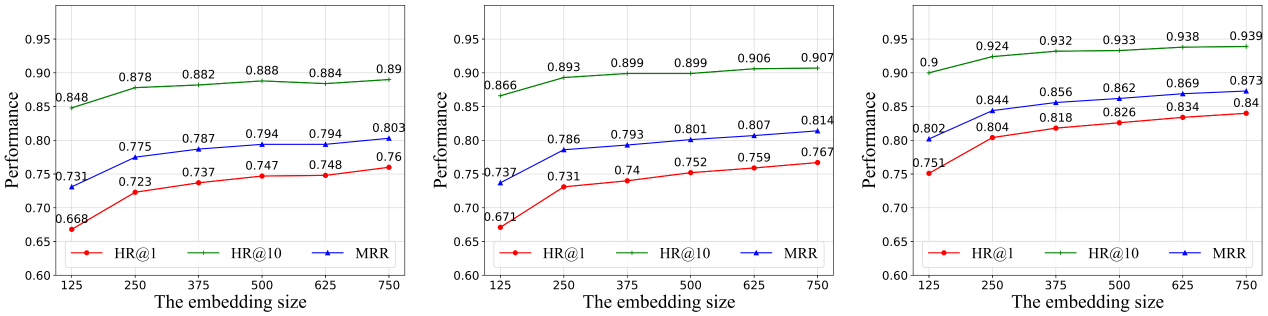
From the comparison result between the RpAlign and wo/self-training, we can found that self-training can achieve about 31.13% at Hits@1 , 9.25% at Hits@10 and 21.40% at MRR. The reason is that self-training technology can help RpAlign learn the embedding of the mixed KG from those un-aligned entities' knowledge triples. It indicates the importance of the unknown information contained in the un-aligned entity pairs. It shows that self-training is a useful method for entity alignment task to deal with the problem of few labeled training data.

The purpose of our model‘s self-training module includes improving three data augmentation methods. Therefore, we studied how the self-training learning mechanism affects the number of triples generated by the data enhancement module. Figure 2a shows the changes in the numbers of triples generated by the three data augmentation methods combined with self-training method. Figure 2b shows the changes in three metrics ranging from 1st to 100th training iteration on DBP$_{ZH-EN}$. It can be seen how self-training module improve the effectiveness of the data augmentation module through the following observation and analysis:

First, We can observe that the number of triples generated by the three data augmentation methods increases except for alignment relations. The self-training module combined with data augmentation technologies continuously adds new aligned entity pairs to the original ones. So it can continuously generate new triples from the third iterations.

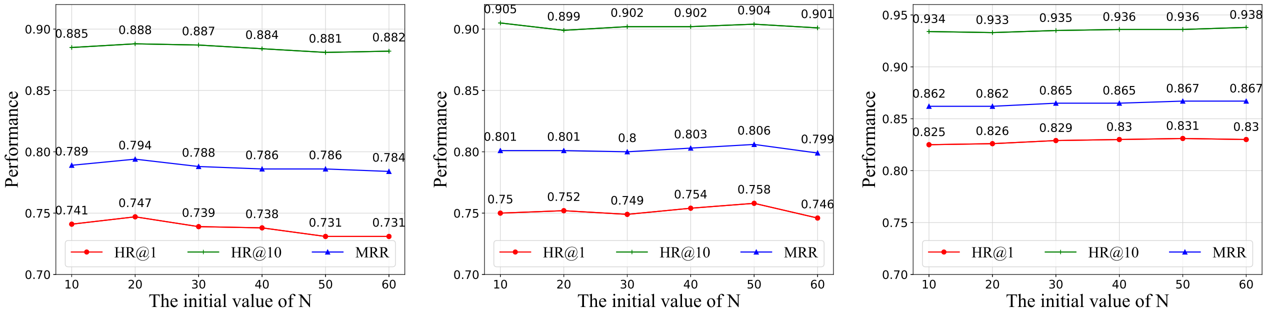
Second, we can observe that the numbers of triples generated from alignment relations are decreasing . Because our model continuously update the probability of alignment entities and relations and selects alignment entities with a high probability of adding to the training data in the self-training phase. The number of new alignment entity pairs increases in the self-training phase, so there are more conflicts in the

[1] Due to the limitation of space, we only present the visualization of impacts of data augmentations on DBP$_{ZH-EN}$, but same results are observed in remaining datasets.



(a) DBP$_{ZH-EN}$ (b) DBP_{FR-EN} (c) $DBP-WD$

Figure 3: The hyper-parameter study of embedding size on DBP$_{ZH-EN}$, DBP$_{FR-EN}$ and DBP-WD datasets. [2]



(a) DBP$_{ZH-EN}$ (b) DBP_{FR-EN} (c) $DBP-WD$

Figure 4: The hyper-parameter study of initial value of N on DBP$_{ZH-EN}$, DBP$_{FR-EN}$ and DBP-WD datasets [2].

alignment relation pairs. Our model eliminates all conflict alignment relation pairs; resulting in a decrease in the number of triples generated form aligned relations. For the other three data augmentation modules, the number of alignment entity pairs added in the self-training phase increases, so the number of triples generated from three data augmentation module is also increasing.

Third, our model's experimental performance would increase with the increase of the total generated triple number, and they are stabilized around almost the same iterations. It means that the total triples generated by RpAlign are very adequate to solve the issue of limited training data. The results indicate the role and effect of three data augmentation techniques combined with the improved self-training method in boosting entity alignment performance.

4.5.3. Impacts of Knowledge embedding module

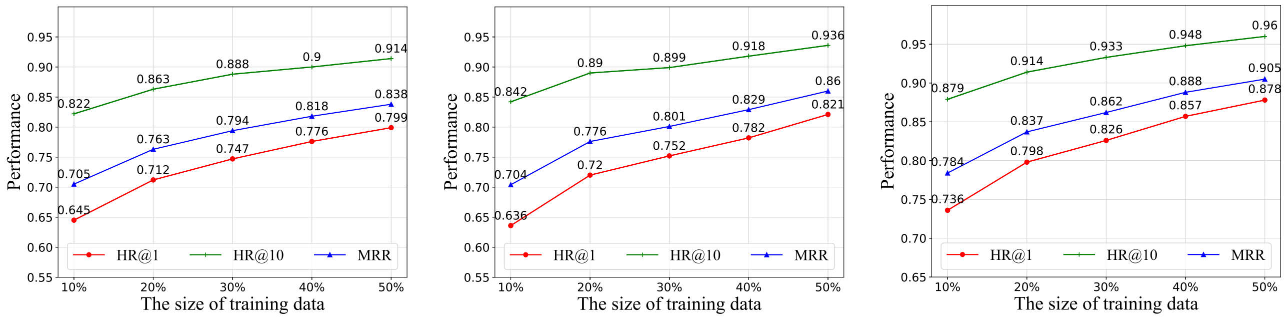
We make ablation study about how the knowledge embedding module affect our model's performance. We can also find that using other KG embedding learning models is not as competitive as using pRotatE. RpAlign can achieve at average 7.20% on Hits@1, 2.94% on Hits@10, and 6.65% on MRR improvement compared with it's variant that choose TransE model as knowledge embedding module. The reasons why we choose pRotatE are following:

First, pRotatE maps the entity of the hybrid KG to the spherical space which are more easier to calibrate. Second, pRotatE regards relations between the entities as the rotation operation between the entities, and it can handle three Kinds of relation patterns: symmetric/antisymmetric, inversion,and composition, while other models cannot handle all three relation patterns, so they are not capable of learning the mixed KG embedding as pRotatE. From the section 3.1.2, our proposed anchor relation is a kind of relation with symmetric/antisymmetric, composition and inversion relation patterns. However, DistMult uses a diagonal matrix to represent the relations, so it cannot effectively handle the inversion relation pattern. ComplEx introduces the complex number extension of DistMult to better model the asymmetric relation pattern, but it cannot effectively model the composition relation pattern. TransE regards the relations to transform the relationship in space and it regards the anchor relation as a zero vector.

4.6. Hyper-parameter Study

In this section, we discuss how the embedding size and N's initial value in the self-training module affect our model's performance. The embedding size is ranged $\{125, 250, 375, 500, 625, 750\}$ for hyper-parameter study experiments, and N's initial value is ranged

[2] Due to the limitation of space, we only present the results of DBP$_{ZH-EN}$, DBP$_{FR-EN}$ and $DBP-WD$ datasets.



(a) DBP$_{ZH-EN}$ (b) DBP_{FR-EN} (c) $DBP-WD$

Figure 5: Visualization of sensitivity study on DBP$_{ZH-EN}$, DBP$_{FR-EN}$ and DBP-WD datasets.

$\{10, 20, 30, 40, 50, 60\}$ at the forth training iteration. The experiments' results are shown in the Fig 3 and Fig 4.

First, we can see from the results of the three data sets that as the embedding size increases, our model's performance also increases. Because as the model's embedding dimension increases, our model's ability to learn the mixed KG's embedding semantics increases.

Second, if N's initial value is too large for the DBP$_{ZH-EN}$ datasets, the performance of the model will decrease instead. Because its number of triples is small, which means the quality of the representation of the mixed KG learned by our model is not enough good to avoid much noisy false aligned entity pairs, and the new predicted false aligned entity pairs introduced by the self-training module would reduces the model's performance. It is necessary to reduce the N's initial value to avoid introducing too much noise for the DBP$_{ZH-EN}$ dataset. For the DBP-WD and DBP$_{FR-EN}$ dataset, the best value of parameter N is larger than DBP$_{ZH-EN}$ dataset. Because the their initial number of training triple is larger than the DBP$_{ZH-EN}$ datasets, and their initial quality of the KG's embedding learned by the model is better. The larger the initial value of N, the more new supervised training data can be introduced to improve its final performance.

4.7. Sensitivity Study

To study the sensitivity of our model to the different initial sizes of prior alignment, we conducted experiments with different proportions, which ranges from 10% to 40% with step 10%. Figure 5 shows the results in Hits@1 and Hits@10 on the three datasets. It can be seen that the performance rises with the increase of proportion, due to the more prior alignment can provide more information. We found that RpAlign could get promising results even only using 10% pre-aligned entities for training data, e.g., Hits@1 are at least 60%, which demonstrates the effectiveness and practicability of RpAlign.

4.8. Case Study

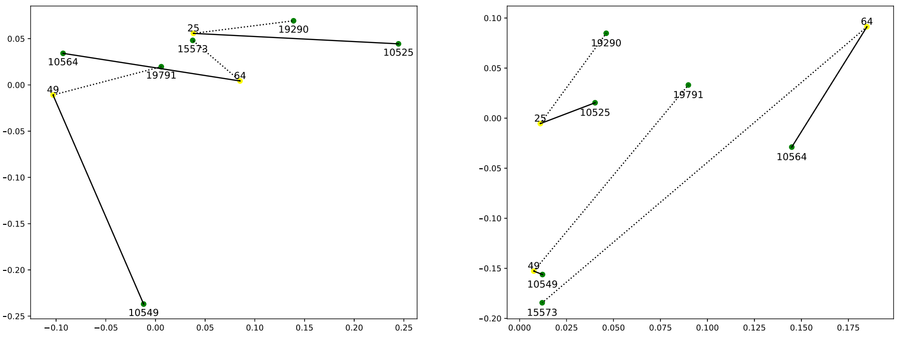
To study the effect of triples from likely aligned relations described in 3.1.1, we chose the unaligned entity pair (Beatrice_Ask$_{FR}$, Beatrice_Ask) in the DBP$_{FR-EN}$ dataset which is predicted wrong by the wo/align-relation model. Table 4 shows the top 5 prediction results for the entity Beatrice_Ask$_{FR}$ predicted by RpAlign and wo/align-relation models. Table 5 shows the number of cross-KG triples generated by RpAlign and wo/align-relation. First, we found that the top 5 entities predicted from wo/align-relation, and the true target entity Beatrice_Ask correspond to Swedish politicians. It demonstrates that the wo/align-relation model was disturbed by similar entities to make the wrong prediction. Second, we found that RpAlign could generate much more triples than wo/align-relation. Moreover, the experimental results demonstrate that these triples can help improve the model's performance. The generated additional triples can supplement missing knowledge information for the aligned entity pair, so RpAlign can learn more accurate embedding for entities and relations. Besides, these noise entities and the real aligned entity have similar original knowledge triple sets, so their original learned entity vectors' semantic information is very close. These additional triples can distinguish the real aligned entity by widening the difference gap between the real and noisy ones.

To study where our model may succeed or fail, we make a case study on some examples on the dataset DBP$_{FR-EN}$. We compute the average triple numbers of the alignment entity pairs in two KGs with our model's different prediction results (successful or fail at Hits@1) in the testing data. And we display them in Table 6. It can be seen from the table that the average triple number of the failure samples is much less than those successful samples. It is consistent with common sense that nodes with more neighbors are easier to align. Take our model's failure predicting alignment entity pair (Reuss$_{FR}$, Reuss_(river)) as the example. There is only one triple for the entity Reuss_(river) in the target KG, and there are seven triples for the entity Reuss$_{FR}$ in the source kG. The entity in the target KG predicted by our model is the entity Alzette, similar to the true entity Reuss_(river) in the target KG. We conclude that as for those alignment entity pairs with fewer triple numbers in the source and target KG, our model is susceptible to interference from similar noisy entities and makes fail predictions.

We also conduct a case study to prove that our model

Table 4
Top5 prediction with probability for entity Beatrice_Ask$_{FR}$ in DBP$_{FR-EN}$.

Model	1	2	3	4	5
RpAlign	Beatrice_Ask(0.99983)	Karin_Enström(7e-05)	Fredrik_Reinfeldt(4e-05)	Tobias_Billström(1e-05)	Carl_Bildt(1e-05)
wo/align-relation	Karin_Enström(0.71535)	Carl_Bildt(0.16774)	Fredrik_Reinfeldt(0.02510)	Tobias_Billström(0.01868)	Cecilia_Malmström(0.01362)
repl.cos	Carl_Bildt(0.10893)	Karin_Enström(0.10819)	Beatrice_Ask(0.10579)	Tobias_Billström(0.10273)	Fredrik_Reinfeldt(0.09677)



(a) AliNet's KG embedding

(b) RpAlign's KG embedding

Figure 6: Visualization of three examples on DBP$_{FR-EN}$ for Alinet and RpAlign. The yellow and green nodes indicate entities in the source and target KG, the number of nodes indicates the entity id in the original data. The solid line represents the correct aligned entity pairs by RpAlign, and the dashed line represents the incorrect ones by AliNet.

Table 5
The external generated cross-KG triples about entity pair (Beatrice_Ask$_{FR}$, Beatrice_Ask).

Model	Beatrice_Ask$_{FR}$	Beatrice_Ask
RpAlign	35	31
wo/align-relation	3	9

Table 6
The statistics about the RpAlign's and AliNet's successful and failure predicting alignment entity pairs on the DBP$_{FR-EN}$.

Model	RpAlign		AliNet	
	successful	Failure	successful	Failure
average triple numbers(source)	13.84	11.45	15.04	13.25
average triple numbers(target)	15.95	9.80	16.89	14.44

can make some successful predictions where other models, like AliNet fail with fewer knowledge triples. First, we also compute the average triple numbers of the alignment entity pairs in two KGs with different prediction results in the testing data from the AliNet. We also show it in Table 6. It can be observed that the average triple numbers from our successful and failure predicting alignment entity pairs are smaller than that of AliNet. So, our model is more robust than AliNet when predicting alignment entity pairs with fewer triple numbers. Second, we chose some cases from the prediction results and give detailed analysis why RpAlign is more robust than AliNet. We selected three testing entities and their successful predicted alignment entities by RpAlign and failure predicted alignment entities by AliNet as case examples. Figure 6 shows their embedding reduction(PCA for dimensionality reduction) representations on two vector spaces learned by AliNet and RpAlign. As shown in the figure, in the vector space of AliNet, the vector distance between successfully predicted entity pair by RpAlign are larger than the failure ones by AliNet. It means that these entities are interfered with by similar entities on AliNet's learned entity embedding. The reason is that the knowledge triple numbers for these entity pairs is small. We take one of the entity pairs (Rugby_Calvisano$_{FR}$, Rugby_Calvisano) to analyse. Their corresponding entity id is 25 and 10525. The AliNet's failure predicted result is (Rugby_Calvisano$_{FR}$, National_Championship_of_Excellence). The entity id of National_Championship_of_Excellence is 19290. The knowledge triple numbers for entity pair (Rugby_Calvisano$_{FR}$, Rugby_Calvisano) is 4 and 5. In addition, the interfering entity National_Championship_of_Excellence and the correct alignment entity both refer to entities in the field of sports events. Therefore, its small amount of knowledge information cannot ensure that similar entities will not interfere with it. However, in the vector space of RpAlign, the wrong alignment entity pairs predicted by AliNet has a vector distance greater than the correct entity pairs. It proves that RpAlign can alleviate the interference of similar entities. The reason is that our data augmentation mechanism and self-training mechanism can generate extra more knowledge triples. For example, our model can generate extra 19 knowledge triples for the entity Rugby_Calvisano$_{FR}$, and 19 knowledge triples for the entity Rugby_Calvisano. These extra triples help RpAlign learn more accurate knowledge embedding for the entity pair to alleviate the interference of similar entities. In conclusion, our model's generated external triples for these entities can effectively alleviate the problem with fewer knowledge triples. So, due to our model's data augmentation and self-training mechanism, our model is more robust than other models like AliNet when predicting alignment entity pairs with fewer knowledge triple numbers.

Our main innovation is to predict the alignment entities based on scores of two entities satisfying the 'anchor' relation instead of their vector distance. We also conduct a case study to prove its strength. We also show the top 5 prediction results of the repl.cos and RpAlign for the entity Beatrice_Ask$_{FR}$ in the source KG on Table 4. The repl.cos is RpAlign's variant that adopts the cosine distance to compute the alignment entity pairs. It can be found that repl.cos is more sensitive to interference from similar entities than RpAlign. Because RpAlign predicts the alignment entities based on the values of two entities satisfying the 'anchor' relation, which can make full use of the entity vector semantic information. So, the example indicates that our main innovation can help reduce the interference of similar entities.

5. Conclusion

This paper presents a novel framework named RpAlign for the entity alignment task. RpAlign defines a new specific 'anchor' relation r_a for aligned entities. It projects two KGs into a unified semantic vector space and uses the link prediction results to predict new aligned entities. RpAlign utilizes data augmentation and the improved self-training method to solve the issue of limited training data. We evaluated RpAlign on the two large real-world datasets, DBP15K and DWY100K, and experimental results show that RpAlign significantly outperforms all baselines. Besides, the analysis of results indicates the effect of each module of RpAlign in improving entity alignment performance. In our future work, we will continue to improve our model. For example, we will adopt a divide-and-conquer algorithm to reduce computing time. We decided to use more entity and relation information to improve our model's conflict resolution mechanism. We also decide to introduce the human-in-the-loop mechanism to improve our model's robustness and performance.

Acknowledgments

We thank all the anonymous reviewers. This work is supported by the National Natural Science Foundation of China (No.61772151 and No.U20B2053).

References

[1] Berrendorf, M., Faerman, E., Tresp, V., 2021. Active learning for entity alignment, in: Hiemstra, D., Moens, M., Mothe, J., Perego, R., Potthast, M., Sebastiani, F. (Eds.), ECIR, Springer. pp. 48–62. URL: https://doi.org/10.1007/978-3-030-72113-8_4, doi:10.1007/978-3-030-72113-8_4.

[2] Bojanowski, P., Grave, E., Joulin, A., Mikolov, T., 2017. Enriching word vectors with subword information. Trans. Assoc. Comput. Linguistics 5, 135–146. URL: https://transacl.org/ojs/index.php/tacl/article/view/999.

[3] Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J., 2008. Freebase: a collaboratively created graph database for structuring human knowledge, in: SIGMOD, AcM. pp. 1247–1250.

[4] Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O., 2013. Translating embeddings for modeling multi-relational data, in: NIPS, pp. 2787–2795.

[5] Cao, Y., Liu, Z., Li, C., Liu, Z., Li, J., Chua, T., 2019. Multi-channel graph neural network for entity alignment, in: ACL, pp. 1452–1461. URL: https://www.aclweb.org/anthology/P19-1140/.

[6] Chen, J., Zhong, M., Li, J., Wang, D., Tu, H., 2021. Effective deep attributed network representation learning with topology adapted smoothing. IEEE Transactions on Cybernetics PP, 1–12.

[7] Chen, M., Tian, Y., Yang, M., Zaniolo, C., 2017. Multilingual knowledge graph embeddings for cross-lingual knowledge alignment, in: IJCAI, pp. 1511–1517. URL: https://doi.org/10.24963/ijcai.2017/209, doi:10.24963/ijcai.2017/209.

[8] Conneau, A., Rinott, R., Lample, G., Williams, A., Bowman, S., Schwenk, H., Stoyanov, V., 2018. Xnli: Evaluating cross-lingual sentence representations, in: EMNLP, pp. 2475–2485.

[9] Cui, Y., Che, W., Liu, T., Qin, B., Wang, S., Hu, G., 2019. Cross-lingual machine reading comprehension, in: EMNLP, pp. 1586–1595. URL: https://doi.org/10.18653/v1/D19-1169, doi:10.18653/v1/D19-1169.

[10] Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S., 2018. Convolutional 2d knowledge graph embeddings, in: AAAI, pp. 1811–1818. URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17366.

[11] Goodfellow, I.J., Bengio, Y., Courville, A.C., 2016. Deep Learning. Adaptive computation and machine learning, MIT Press. URL: http://www.deeplearningbook.org/.

[12] Hu, S., Zou, L., Yu, J.X., Wang, H., Zhao, D., 2017. Answering natural language questions by subgraph matching over knowledge graphs. IEEE Transactions on Knowledge and Data Engineering 30, 824–837.

[13] Hu, W., Chen, J., Qu, Y., 2011. A self-training approach for resolving object coreference on the semantic web, in: WWW, ACM. pp. 87–96.

[14] Ji, G., He, S., Xu, L., Liu, K., Zhao, J., 2015. Knowledge graph embedding via dynamic mapping matrix, in: ACL, pp. 687–696. URL: https://www.aclweb.org/anthology/P15-1067/.

[15] Kipf, T.N., Welling, M., 2017. Semi-supervised classification with graph convolutional networks, in: ICLR. URL: https://openreview.net/forum?id=SJU4ayYgl.

[16] Lample, G., Conneau, A., Ranzato, M., Denoyer, L., Jégou, H., 2018. Word translation without parallel data, in: ICLR. URL: https://openreview.net/forum?id=H196sainb.

[17] Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., Van Kleef, P., Auer, S., et al., 2015. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. Semantic Web 6, 167–195.

[18] Li, C., Wang, S., Wang, Y., Yu, P.S., Liang, Y., Liu, Y., Li, Z., 2019. Adversarial learning for weakly-supervised social network alignment, in: AAAI, pp. 996–1003. URL: https://doi.org/10.1609/aaai.v33i01.3301996, doi:10.1609/aaai.v33i01.3301996.

[19] Li, Z., Wang, X., Li, J., Zhang, Q., 2021. Deep attributed network representation learning of complex coupling and interaction. Knowl. Based Syst. 212, 106618. URL: https://doi.org/10.1016/j.knosys.2020.106618, doi:10.1016/j.knosys.2020.106618.

[20] Lin, Y., Liu, Z., Luan, H., Sun, M., Rao, S., Liu, S., 2015a. Modeling relation paths for representation learning of knowledge bases, in: EMNLP, pp. 705–714. URL: https://doi.org/10.18653/v1/d15-1082, doi:10.18653/v1/d15-1082.

[21] Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X., 2015b. Learning entity and relation embeddings for knowledge graph completion, in: AAAI, pp. 2181–2187. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9571.

[22] Liu, J., Lin, Y., Liu, Z., Sun, M., 2019. Xqa: A cross-lingual open-domain question answering dataset, in: ACL, pp. 2358–2368.

[23] McClosky, D., Charniak, E., Johnson, M., 2008. When is self-training effective for parsing?, in: COLING, pp. 561–568. URL: https://www.aclweb.org/anthology/C08-1071/.

[24] Noy, N.F., Musen, M.A., 2000. PROMPT: algorithm and tool for automated ontology merging and alignment, in: Proceedings of the Sev-

enteenth National Conference on Artificial Intelligence and Twelfth Conference on on Innovative Applications of Artificial Intelligence, July 30 - August 3, 2000, Austin, Texas, USA, pp. 450–455. URL: http://www.aaai.org/Library/AAAI/2000/aaai00-069.php.

[25] Noy, N.F., Musen, M.A., 2001. Anchor-prompt: Using non-local context for semantic matching, in: IJCAI-01. URL: http://ceur-ws.org/Vol-47/noy.pdf.

[26] Peng, H., Li, J., Gong, Q., Song, Y., Ning, Y., Lai, K., Yu, P.S., 2019. Fine-grained event categorization with heterogeneous graph convolutional networks, in: Proceedings of the 28th International Joint Conference on Artificial Intelligence, AAAI Press. pp. 3238–3245.

[27] Shorten, C., Khoshgoftaar, T.M., 2019. A survey on image data augmentation for deep learning. J. Big Data 6, 60. URL: https://doi.org/10.1186/s40537-019-0197-0, doi:10.1186/s40537-019-0197-0.

[28] Suchanek, F.M., Kasneci, G., Weikum, G., 2007. Yago: a core of semantic knowledge, in: WWW, ACM. pp. 697–706.

[29] Sun, J., Zhou, Y., Zong, C., 2020a. Dual attention network for cross-lingual entity alignment, in: COLING, pp. 3190–3201. URL: https://doi.org/10.18653/v1/2020.coling-main.284, doi:10.18653/v1/2020.coling-main.284.

[30] Sun, Z., Deng, Z., Nie, J., Tang, J., 2019. Rotate: Knowledge graph embedding by relational rotation in complex space, in: ICLR. URL: https://openreview.net/forum?id=HkgEQnRqYQ.

[31] Sun, Z., Hu, W., Li, C., 2017. Cross-lingual entity alignment via joint attribute-preserving embedding, in: ISWC, Springer. pp. 628–644.

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

[33] Sun, Z., Wang, C., Hu, W., Chen, M., Dai, J., Zhang, W., Qu, Y., 2020b. Knowledge graph alignment network with gated multi-hop neighborhood aggregation, in: AAAI, pp. 222–229. URL: https://aaai.org/ojs/index.php/AAAI/article/view/5354.

[34] Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G., 2016. Complex embeddings for simple link prediction, in: ICML, pp. 2071–2080.

[35] Tuan, Y.L., Chen, Y.N., Lee, H.y., 2019. Dykgchat: Benchmarking dialogue generation grounding on dynamic knowledge graphs, in: EMNLP, pp. 1855–1865.

[36] Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y., 2018. Graph attention networks, in: ICLR. URL: https://openreview.net/forum?id=rJXMpikCZ.

[37] Wang, H., Zhang, F., Wang, J., Zhao, M., Li, W., Xie, X., Guo, M., 2018a. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems, in: CIKM, pp. 417–426. URL: https://doi.org/10.1145/3269206.3271739, doi:10.1145/3269206.3271739.

[38] Wang, Z., Li, J., Tang, J., 2013. Boosting cross-lingual knowledge linking via concept annotation, in: IJCAI.

[39] Wang, Z., Lv, Q., Lan, X., Zhang, Y., 2018b. Cross-lingual knowledge graph alignment via graph convolutional networks, in: EMNLP, pp. 349–357.

[40] Wang, Z., Zhang, J., Feng, J., Chen, Z., 2014. Knowledge graph embedding by translating on hyperplanes, in: AAAI, pp. 1112–1119. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8531.

[41] Xiang, Y., Zhang, Z., Chen, J., Chen, X., Lin, Z., Zheng, Y., 2021. Ontoea: Ontology-guided entity alignment via joint knowledge graph embedding, in: ACL/IJCNLP, pp. 1117–1128. URL: https://doi.org/10.18653/v1/2021.findings-acl.96, doi:10.18653/v1/2021.findings-acl.96.

[42] Xie, Y., Li, C., Yu, B., Zhang, C., Tang, Z., 2020. A survey on dynamic network embedding. CoRR abs/2006.08093. URL: https://arxiv.org/abs/2006.08093, arXiv:2006.08093.

[43] Xu, K., Wang, L., Yu, M., Feng, Y., Song, Y., Wang, Z., Yu, D., 2019. Cross-lingual knowledge graph alignment via graph matching neural network, in: ACL, pp. 3156–3161. URL: https://doi.org/10.18653/v1/p19-1304, doi:10.18653/v1/p19-1304.

[44] Yang, B., Yih, W., He, X., Gao, J., Deng, L., 2015. Embedding entities and relations for learning and inference in knowledge bases, in: ICLR. URL: http://arxiv.org/abs/1412.6575.

[45] Ye, R., Li, X., Fang, Y., Zang, H., Wang, M., 2019. A vectorized relational graph convolutional network for multi-relational network alignment, in: IJCAI, pp. 4135–4141.

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

[47] Zhu, Q., Zhou, X., Wu, J., Tan, J., Guo, L., 2019. Neighborhood-aware attentional representation for multilingual knowledge graphs, in: IJCAI, pp. 1943–1949.