SAEA: Self-Attentive Heterogeneous Sequence Learning Model for Entity Alignment

No Author Given

No Institute Given

Abstract. We consider the problem of entity alignment in knowledge graphs. Previous works mainly focus on two aspects: One is to improve the TransE-based models which mostly only consider triple-level structural information i.e. relation triples or to make use of graph convolutional networks holding the assumption that equivalent entities are usually neighbored by some other equivalent entities. The other is to incorporate external features, such as attributes types, attribute values, entity names and descriptions to enhance the original relational model. However, the long-term structural dependencies between entities have not been exploited well enough and sometimes external resources are incomplete and unavailable. These will impair the accuracy and robustness of combinational models that use relations and other types of information, especially when iteration is performed. To better explore structural information between entities, we novelly propose a Self-Attentive heterogeneous sequence learning model for Entity Alignment (SAEA) that allows us to capture long-term structural dependencies within entities. Furthermore, considering low-degree entities and relations appear much less in sequences prodeced by traditional random walk methods, we design a degree-aware random walk to generate heterogeneous sequential data for self-attentive learning. To evaluate our proposed model, we conduct extensive experiments on real-world datasets. The experimental results show that our method outperforms various state-of-the-art entity alignment models using relation triples only.

Keywords: Knowledge Graph · Entity Alignment · Degree-aware Random Walk

1 Introduction

Knowledge graphs (KGs) are playing an increasingly important role in many basic applications of artificial intelligence, like question answering and recommendation systems. Many KGs including the most popular ones such as DBpedia [10], YAGO [16] and Freebase [2] are created by different methods in different languages, which makes them inevitably heterogeneous. Therefore, Entity alignment is proposed aiming to find entities in different KGs referring to the same real-world object.

Conventional methods [12, 15, 21] for entity alignment mainly rely on string similarities, such as entity names and attribute values. Nevertheless, the computation cost increases exponentially with the increase in the number of entities in

KGs, and when it comes to a cross-lingual scenario, symbolic features are hard to be extracted. Recently, many methods leverage KG embedding techniques to address the above problems. Their key idea is to encode entities and relations into semantic space and find alignment between entities according to their embedding similarities in this space. TransE [3] is the most popular one in the KG embedding area which makes each relation triple (h, r, t) meet the requirement of $\mathbf{h}+\mathbf{r}\approx\mathbf{t}$ in vector space. For instance, MTransE [5] utilizes TransE to learn vector representations of entities in different KGs in separated embedding space and then maps each embedding vector of seed alignments to its corresponding counterpart via different transitions. However, the number of seed alignments is limited. SEA [13] is then proposed, leverages both seed alignments and plenty of unlabeled entities in a semi-supervised way and alleviates the effect of entity's degree differences in TransE-based KG embedding methods. The above two alignment models can be noted as TransE-based methods which are known as triple-level learning. It is obvious that there exists low expressiveness and inefficient information propagation in triple-level learning as it learns entity embeddings from triples and disseminates alignment information employing limited seed entities.

Many other models use iteration and extra resources to improve alignment results. IPTransE [26] and BootEA [18] design elaborate iteration strategies to acquire more high-confidence aligned entities for training. JAPE [17], AttrE [19], GCN-Align [22] and MultiKE [25] jointly model structure and external resources like attributes and names to refine entity embeddings. However, the above methods are still TransE-based methods where the potential of structural semantics has not been exploited well enough, not to mention that sometimes external information is incomplete and unreliable or unavailable, making models that combining external resources not applicable in some scenarios.

To better explore structural information between entities, RSNs [7] is proposed focusing on learning from relational paths which are composed of entities and relations alternately and have achieved state-of-the-art performance using relation triples only. However, its RNN-based sequence learning model is **difficult to learn dependencies between distant positions** as the output at time step t is determined by the current input and a mix of information of all the previous inputs. Thus, it cannot capture long-term dependencies between entities effectively and efficiently. Beyond that, when making use of random walk to sample relational paths, it ignores that random walk is **biased to entities of high degree** which results in extremely uneven information collection between long-tail entities and norm entities.

Considering limitations of the above method, we believe it is of critical importance to develop a model that can not only capture long-term dependencies between entities in an efficient manner but also lay more emphasis on low-degree entites. Towards this end, inspired by the new sequential model Transformer [20], which has achieved better performance than traditional recurrent models in machine translation tasks but have not been explored in KGs for entity alignment, we propose a brand-new **S**elf-**A**ttentive heterogeneous sequence learning model

for **E**ntity **A**lignment (**SAEA**). The key idea of SAEA is to model dependencies without regard to the distance between items in the sequence by solely using self-attention mechanisms. However, it is necessary to customize due to the heterogeneity in sequences. Furthermore, unlike RNN-based models assuming that the next element in a relational path depends on the current input and hidden state which is inappropriate for paths in KGs, we adapt the original residual connection in Transformer to a special crossed residual module.

In addition, to generate ideal relational paths for sequence learning, we design a **degree-aware random walk**, which pays more attention to low-degree entities. It differs from previous random walks in the network embedding area in two points. One is that the sampled path is made up of two kinds of nodes—entities and relations. The other is we control the biases according to the frequency of relations and degree of entities. The output of the degree-aware random walk is then sent to our heterogeneous sequence learning model to learn embeddings. After sequence learning, we get alignment results via calculating embedding similarities.

Our main contributions in this paper are summarized as follows:

- We propose a self-attentive model for entity alignment. To the best of our knowledge, we are the first to manage to apply self-attention mechanisms to heterogeneous sequences in KGs for alignment.
- We also propose to generate heterogeneous sequences in KGs with a designed degree-aware random walk.
- We conduct extensive experiments on four real-world datasets and the result demonstrates our model outperforms TransE-based methods significantly and achieves state-of-the-art performance.

Roadmap. The rest of the paper is organized as follows: We discuss the related works in Sec. 2. After state the problem definition in Sec. 3, our proposed method is presented in Sec. 4 and followed by reporting our empirical study in Sec. 5. We finally conclude the whole paper in Sec. 6.

2 Related Work

Entity alignment is a subtask of ontology alignment consisting of schema and instance matching. In this section, we first discuss the existing methods for entity alignment from two aspects and then introduce sequence learning which is at the core of our model.

2.1 Entity Alignment

According to the number of resources used, the existing entity alignment models fall into two categories.

Single-Resource-based. In most cases, entity alignment can achieve high accuracy by simply comparing string similarities between entity names. However, as the number of entities increases, the computation cost is very high, not to

mention that sometimes names are unavailable and difficult to compare in cross-lingual scenarios. In the past few years, much work has been done on the problem of KG embedding. As the entities can be represented by vectors, entity alignment can be implemented by comparing the distance between vectors. MTransE [5] encodes relation triples by utilizing TransE method and provides transitions for entity embeddings in one KG to map into the counterpart entity in the other KG. The loss function of MTransE is the weighted sum of TransE and alignment model. GCN-Align [22] employs graph convolutional networks (GCNS) to embed entities based on adjacent entities linked by different relations. To train the above two alignment models, a set of aligned entities between KGs are needed, leaving the abundant unaligned entities without consideration. SEA [13] is targeted at designing a semi-supervised entity alignment model learning from both aligned and unaligned entities rather than IPTransE [26] and BootEA [18] focus on acquiring more aligned entities using iterative strategy. These models use single-resource—structural information between entities.

Multi-Resource-based. In addition to using structural information, many works leverage extra resources, such as OWL properties [8], entity descriptions [24] and attribute information of entities. JAPE [17] jointly embeds the structures of two KGs into a unified vector space and further refines it by leveraging attribute correlations in entities. KDcoE [4] iteratively trains two components embedding models on multilingual KG structures and entity descriptions respectively using co-training strategy. AttrE [19] generates attribute character embeddings for large numbers of attribute triples to shift the entity embeddings from two KGs into the same space by computing the similarity between entities based on their attributes. MultiKE [25] uses three different strategies to combine three representative views—name, relation and attribute features each of which uses an appropriate model to learn entity embeddings. Such methods are usually limited by the availability of the extra information in KGs. Therefore, we still aim at learning from structural information between entities as it is the intrinsic feature and stable. However, different from previous single-resource-based models which are hard to propagate alignment information effectively, our approach is capable of learning embeddings through relational paths that containing richer information than triples and neighbors.

2.2 Sequence Learning

In recent years, numerous works try to apply sequence models to graph structures, as sequence learning has been widely used in many fields, for instance, machine translation, speech recognition and music generation, and achieved excellent performance on many tasks. However, in the network and KG embedding area, inputs are naturally represented as graphs rather than sequence, thus existing Seq2Seq models [23] face a significant challenge in achieving appropriate sequence from graphs. DeepWalk [14] is the pioneer of generating sequences in graph. It introduces uniform random walk to sample paths while node2vec [6] uses biased random walk to explore nodes in a breadth-first-search and depth-first-search fashion. In this paper, we design a degree-aware random walk in KGs

concentrating on low-degree entities and relations of low frequency in case the vast majority of sampled paths point to a small set of high-frequent entities. Besides, although the dominant sequence models are based on combining complicated recurrent or convolutional networks with attention mechanisms, their inherent sequential nature precludes parallelization. Therefore, stimulated by the Transformer [20], a purely self-attention based sequence model achieving the start-of-the-art performance and efficiency, we seek to build a sequential alignment model based upon it. The model needs to be specially designed due to the differences between entity alignment and traditional sequence tasks.

3 Probelm Definition

KGs represent knowledge about the real-world entities as triples. It is a directed multi-relational graph where nodes denote entities and edges have directions and labels indicate that there exist a specific relation from one entity to the other.

Formally, a KG can be noted as $KG = \{E, R, T\}$, where E is the set of entities, R is the set of relations, and T is the set of triples, each of which is a triple (h, r, t), including the head entity h, the relation r and the tail entity t. Each triple can be presented as $(\mathbf{h}, \mathbf{r}, \mathbf{t})$ by KG embedding techniques in which $\mathbf{h}$, $\mathbf{r}$, $\mathbf{t}$ represent the embeddings of head entity h, relation r and tail entity t, respectively.

Given two KGs: $KG_1 = (E_1, R_1, T_1)$, $KG_2 = (E_2, R_2, T_2)$ and a set of pre-aligned entities $AS = \{(e_1^i, e_2^j) | e_1^i \in E_1^a, e_2^j \in E_2^a\}$ between KG_1 and KG_2 where E_1^a and E_2^a are subsets of E_1 and E_2 respectively, entity alignment is a task to find the remaining semantically same entity set $US = \{(e_1^i, e_2^j) | e_1^i \in E_1^u, e_2^j \in E_2^u\}$ where $E_1^u = E_1 \setminus E_1^a$ and $E_2^u = E_2 \setminus E_2^a$.

4 Our Approach

The framework of our proposed method is shown in Fig. 1: The whole process can be divided into two phases where the sequence generating phase is the prerequisite of the sequence learning phase. In phase one, the input layer is composed of two KGs and a set of known aligned entity pairs between them. The basic idea of our approach is to utilize an appropriate random walk method to generate heterogeneous sequences from KGs so that sequence learning model is allowed to be applied in our situation. After the well-designed phase two, entity alignment is predicted by measuring the cosine similarity in the embedding space.

In the following of this section, we start with degree-aware random walk for sequence generating in Sec. 4.1. Then, in Sec. 4.2. we describe self-attention based sequence learning for entity alignment in detail.

4.1 Sequence Generating with Random Walk

Random walk is the most basic way to learn node embeddings in the network embedding area by generating sequences. Their core innovation is to optimize the

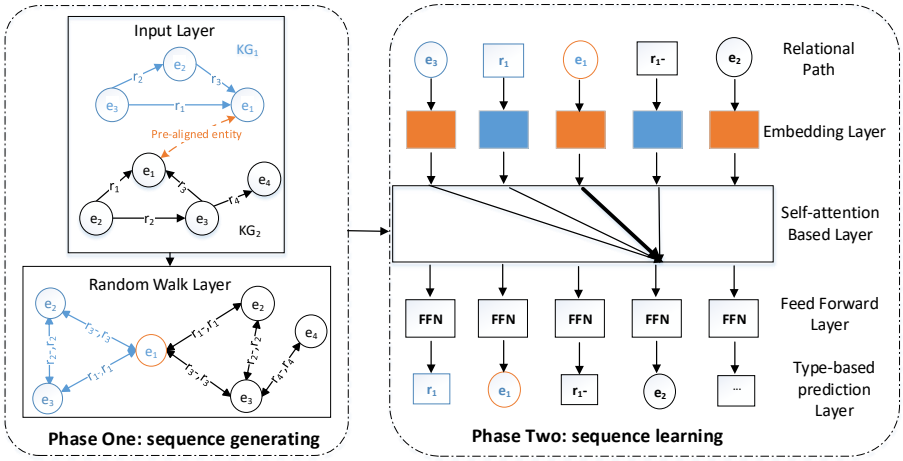

Fig. 1. Framework of our approach

node embeddings so that nodes have similar embeddings if they tend to co-occur on short random walk paths over the graph. Towards KG embeddings, the edges between nodes have labels and directions, making it different from traditional network embeddings and, for entity alignment, there are two KGs for sampling paths in which the information may be complementary. Inspired by node2vec able to trade off between embeddings that emphasize local structural roles and embeddings that emphasize community structures, we propose a degree-aware random walk between KGs, which allows for deep relational paths and is sensitive to the degree of nodes at the same time.

Traditional Random Walk. We first clarify that the sampled path in KGs is a cross composition of entities and relations which means it has two types of nodes. When applying traditional random walk node2vec to KGs, it follows the equation below to calculate the probability of passing the next entity:

$$P(e_{i+1}|e_i) = \begin{cases} \alpha_{pq}(e_i, e_{i+1}) \cdot w_r & \exists r \in R, (e_i, r, e_{i+1}) \in T \\ 0 & otherwise \end{cases} \tag{1}$$

where e_i is the i^{th} entity in this walk. The walk now needs to decide on the next entity so it evaluates the transition probility from e_i to e_{i+1} if there is a relation r between them. w_r is the edge weight between entity e_i and e_{i+1} and $\alpha_{pq}(e_i, e_{i+1})$ is computed as follows by introducing two random walk hyperparameters p and q:

$$\alpha_{pq}(e_i, e_{i+1}) = \begin{cases} \frac{1}{p} & \text{if } d_{e_{i-1} \to e_{i+1}} = 0 \\ 1 & \text{if } d_{e_{i-1} \to e_{i+1}} = 1 \\ \frac{1}{q} & \text{if } d_{e_{i-1} \to e_{i+1}} = 2 \end{cases} \tag{2}$$

where e_{i-1} denotes the previous entity of e_i and $d_{e_{i-1} \to e_{i+1}}$ represents the shortest path distance between entity e_{i-1} and e_{i+1}. Note that it is obvious that $\{0, 1, 2\}$ covers all situations.

Algorithm 1: Degree-aware Random Walk

Input : $KG_1 = (E_1, R_1, T_1)$, $KG_2 = (E_2, R_2, T_2)$ and a set of pre-aligned entities $AS = \{(e_1^i, e_2^j)|e_1^i \in E_1^a, e_2^j \in E_2^a\}$, bias q, path length l, sampling times t.

Output: Paths containing entities and relations.

1. Combine two KGs as one $KG = (E, R, T)$;
2. Calculate transition probability distribution $P(e_{i+1}|e_i)$ for every couple of r and e_{i+1} if $\exists r \in R, (e_i, r, e_{i+1}) \in T$;
3. **for** i = *1 to* t **do**
   4. **for** *each* $(e_i, r, e_{i+1}) \in T$ **do**
      5. $p = e_i \rightarrow r \rightarrow e_{i+1}$;
      6. **while** $length(p) < l$ **do**
         7. find the most likely couple of r' and e' in T followed by the last entity in p according to the probability;
         8. $p = p \rightarrow r' \rightarrow e'$;
      end
   end
end
return p;

Degree-aware Random Walk. In the entity alignment task, there are two graphs to sample paths. Considering that they have pre-aligned entities linked with each other, we first combine them as one joint graph by swapping aligned-entities in their triples and add reverse relations between entities to enhance connectivity. Second, entities and relations in KGs have different degrees, to solve the problem of path imbalance caused by this, we introduce degree-aware bias to favor long-tail entities and relations. Formally, the degree-aware bias between e_i and e_{i+1}, denoted by $w_{e_i \rightarrow e_{i+1}}$, is defined as follows:

$$w_{e_i \rightarrow e_{i+1}} = \begin{cases} \frac{1}{d_{e_{i+1}} + f_r} & \exists r \in R, (e_i, r, e_{i+1}) \in T \\ 0 & otherwise \end{cases} \tag{3}$$

where $d_{e_{i+1}}$ is the degree of entity e_{i+1} which indicates the number of entities connected with e_{i+1} and f_r refers to the frequency of relation r that connecting entity e_i and e_{i+1}.

Besides, to prevent the path from walking backward, we leverage the idea of node2vec to control the behavior of random walk to generate deep and acyclic paths. We define the following function to reach our goal:

$$\alpha_q = \begin{cases} q & \text{if } d_{e_{i-1} \rightarrow e_{i+1}} = 2 \\ 1 - q & \text{if } d_{e_{i-1} \rightarrow e_{i+1}} \neq 2 \end{cases} \tag{4}$$

The overall transition probability distribution of next entity can be achieved by repalcing $\alpha_{pq}(e_i, e_{i+1})$ and w_r in equation (1) with α_q and $w_{e_i \rightarrow e_{i+1}}$ respectively. The complete degree-aware random walk process is shown in Algorithm 1.

4.2 Sequence Learning with Self-attention Mechanism

In the setting of sequence learning for entity alignment, we are given a heterogeneous sequence $p = (e_1, r_1, e_2, r_2, ..., e_l)$, and seek to predict the next item based on the previous items. As depicted in phase two in Fig. 1, it is easy to see that the model's output is a shifted version of the input sequence. In this subsection, we demonstrate how the sequential model operates to capture the long-term dependencies between entities via three components.

Embedding Layer. As there are two types of nodes in the path, we create an entity embedding matrix $\boldsymbol{E}$ and a relation embedding matrix $\boldsymbol{R}$. A learnable position embedding $\boldsymbol{P}$ also needs adding to the input embeddings since the self-attention mechanism is not aware of the positions. Hence, the embedding layer can be denoted as $\hat{E}$, as follows:

$$\hat{E} = \begin{bmatrix} \boldsymbol{E}_1 + \boldsymbol{P}_1 \\ \boldsymbol{R}_1 + \boldsymbol{P}_2 \\ \boldsymbol{E}_2 + \boldsymbol{P}_3 \\ ... \\ \boldsymbol{E}_l + \boldsymbol{P}_{2l-1} \end{bmatrix} \tag{5}$$

Self-attention Block. This module is made up of self-attention based layer and feed forward layer, and can be stacked to learn more complex features.

We start with the self-attention based layer by introducing the most commonly used scaled dot-product attention, defined as:

$$Attention(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = softmax(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d}})\boldsymbol{V} \tag{6}$$

where $\boldsymbol{Q}, \boldsymbol{K}$ and $\boldsymbol{V}$ represent queries, keys and values respectively. The weighted sum computed by queries and keys is then assigned to values as the final output. To counteract the extremely small gradients caused by large values of dimension, $\frac{1}{\sqrt{d}}$ is implemented to scale the dot products. The vast majority of attention in sequence learning are used with $\boldsymbol{K} = \boldsymbol{V} \neq \boldsymbol{Q}$.

However, the self-attention mechanism makes queries, keys and values the same. In our scenario, take embedding $\hat{E}$ as input, we find it is beneficial to concatenate h self-attention functions whose input is projected versions of queries, keys and values. The equation is defined as:

$$\begin{aligned} S = SA(\hat{E}) = MultiHead(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = Concat(head_1, ..., head_h) \\ where\, head_i = Attention(\hat{E}W_i^Q, \hat{E}W_i^K, \hat{E}W_i^V) \end{aligned} \tag{7}$$

Where W_i^Q, W_i^K and W_i^V are projection matrices. This multi-head attention allows the model to learn representations independently and is capable of preventing overfitting by ensembling different attentions.

Owing to the uniqueness of paths in KGs that sampled in units of triples and the entanglements between the embedding of last visited item and all previous items after several self-attention blocks, instead of adding common residual

connections to propagate the last item's features, we hold the view that when the next item is entity, the last entity is the most important feature and use a crossed residual connection (C) to realize it. This operation is simply omitted in the figure, but its visual display is the role of the thick arrow in the center of phase two. It is calculated with the following equation:

$$C_i = \begin{cases} S_i & i = odd \\ S_i + \hat{E}_{\frac{i}{2}} & i = even \end{cases} \tag{8}$$

where i denotes the i^{th} item in the relational path.

Although through the self-attention and crossed residual function, the model is capable of aggregating features of all previous items and paying more attention to some specific items, it is unable to interact with different latent dimensions because it is essentially a linear model. Hence, a two-layer feed-forward network is applied to each position separately and identically:

$$F_i = FFN(C_i) = ReLU(C_i W_1 + b_1)W_2 + b_2 \tag{9}$$

where W_1 and W_2 are weight matrices, b_1 and b_2 are bias vectors and $ReLU$ is activation function.

The above three steps form one self-attention block. Though its output is an aggregation of all items which means it can capture long-term dependencies effectively, it is encouraging to stack more blocks to learn more complex item transitions and boost the performance. The b^{th} block is defined as:

$$S^b = SA(F^{b-1}) \tag{10}$$

where the first block is denoted as $S^0 = S$ and $F^0 = F$.

However, simply stacking more blocks suffers from overfitting and unstable training process due to more parameters. Inspired by [20], we adopt layer normalization and dropout strategy in the self-attention layer and feed-forward layer to stabilize and accelerate the model. The entire formulas for them are shown as follows:

$$\begin{aligned} SA(x) &= Dropout(SA(LayerNorm(x))) \\ FFN(x) &= x + Dropout(FFN(LayerNorm(x))) \end{aligned} \tag{11}$$

Type-based Prediction Layer. After stacking b self-attention blocks, we predict the next item based on previous items. As we can see, different from common sequence prediction tasks, we have two kinds of elements in our relational path, so it is simple and convenient to separate them and that is why we call our last layer type-based prediction layer. Besides, KGs usually have large amounts of entities and relations making every prediction time-consuming. To deal with the difficulty of having too many output vectors that need to be updated every epoch, we use negative sampling [11] to update a sample of them.

It is obvious that the output item should be kept in our sample, and we need to sample a few entities or relations as negative samples according to the

Table 1. Statistics of datasets

Datasets	Source KGs	#Entity	#RelationN	#TripleN	#RelationD	#TripleD
DBP-WD	DBPdeia (English)	15,000	253	38,421	220	68,598
	Wikidata (English)	15,000	144	40,159	135	75,465
DBP-YG	DBPdeia (English)	15,000	219	33,571	206	71,257
	YAGO3 (English)	15,000	30	34,660	30	97,131
EN-FR	DBPdeia (English)	15,000	221	36,508	217	71,929
	DBPdeia (French)	15,000	177	33,532	174	66,760
EN-DE	DBPdeia (English)	15,000	225	38,281	207	56,983
	DBPdeia (German)	15,000	118	37,069	117	59,848

predicted item is an entity or a relation. A specific probabilistic distribution is needed for the sampling process to generate suitable negative samples. In our case, for each positive entity, we generate entities whose frequency is in the first three-quarters of the overall frequency distribution as its negative samples, and the positive relations are handled in the same way. The overall loss is defined as follows:

$$\begin{aligned} L = \sum_{i_e=1}^{p_e} \Big(-\log \sigma(F_{i_e} \cdot y_{i_e}) - \sum_{j_e=1}^{n_e} \big(\log \sigma(-F_{i_e} \cdot y_{j_e})\big)\Big) + \\ \sum_{i_r=1}^{p_r} \Big(-\log \sigma(F_{i_r} \cdot y_{i_r}) - \sum_{j_r=1}^{n_r} \big(\log \sigma(-F_{i_r} \cdot y_{j_r})\big)\Big) \end{aligned} \tag{12}$$

where σ is the sigmoid activation function, F_{i_e} is the embedding of output entity, y_{i_e} is its label, p_e and n_e represent the number of positive and negative samples respectively. When the subscript is r, the prediction process of r is explained.

5 Experiments

In this section, we conduct experiments on four couples of real-world datasets with different entity distributions and evaluate our proposed model on them with several metrics.

5.1 Datasets

We use DBPedia, Wikidata and YAGO3 datasets in the experiment, which were built by [7]. The datasets contain English (EN), French (FR) and German (DE) knowledge graphs. To comprehensively evaluate the effectiveness of entity alignment models, the distribution of entities in those sampled datasets are guaranteed to follow the original KGs. Table 1 outlines the detail information of the datasets. Each dataset has a norm and a dense version. RelationN and TripleN represent the number of relation and triple in norm datasets while RelationD and TripleD represent that in dense datasets. However, whether the dataset is norm or dense, all KGs contain 15,000 entities.

5.2 Experiment Settings

As for the evaluation metrics, we adopt Hit@k and MRR to assess the performance of all the approaches. Hit@k measures the proportion of correctly aligned entities ranked in the top k candidates. MRR (Mean Reciprocal Rank) calculates the average ranking scores of the aligned entities whose score is the reciprocal of its rank. Both metrics are preferred to be higher to represent better performance.

For all the compared approaches, we follow the previous works using 30% of pre-aligned entities for training and the rest 70% for testing. The split and dimension of knowledge graph embeddings are the same for all methods.

For the parameters of our approach, we set dimension = 256, learning rate =0.003, dropout = 0.5, q = 0.9, l=15 and t=2. The optimizer is the Adam optimizer, the number of self-attention block and head is 2 and 8 respectively.

5.3 Comparative Methods

To show the effectiveness of our model, we picked up two types of state-of-the-art entity alignment models for comparison.

The first type is single-resource-based models which only consider structural information without extra resources.

- **MTransE**: This a simple baseline that finds alignment via a unidirectional transition matrix between the embeddings of KGs which is based on TransE.
- **SEA**: A novel semi-supervised model for alignment. SEA solves the problem existing in the present embedding methods which is caused by the degree difference of entities in different KGs. A cycled consistent loss is used where bidirectional transition matrices are trained to incorporate unlabeled entities.
- **RSNs**: A state-of-the-art entity alignment method that leverages recurrent neural networks to model relational paths instead of triples to capture long-term dependencies between entities and relations.
- **IPTranE**: An iterative version of the TransE model which jointly encodes both entities and relations of various KGs into a unified low-dimension semantic space.
- **BootEA**: A bootstrapping approach to embedding-based entity alignment. It uses a truncated uniform negative sampling method in the embedding phase and employs an alignment editing method to reduce error accumulation during iterations.

The second type is multi-resource-based approaches, which not only consider relation triples but also attribute triples to some extent.

- **JAPE**: This is the first model to learn embeddings of cross-lingual KGs while preserving their attribute information. By leveraging the attribute triples of KGs with attribute embedding, the original structural embeddings of entities can be refined.

Table 2. Results on norm and monolingual datasets

Datasets	DBP-WD			DBP-YG		
Metrics	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR
MTransE	22.3	50.1	0.32	24.6	54.0	0.34
SEA	31.0	51.9	0.38	30.3	45.9	0.36
RSNs	38.8	65.7	0.49	40.0	67.5	0.50
IPTransE	23.1	51.7	0.33	22.7	50.0	0.32
BootEA	32.3	63.1	0.42	31.3	62.5	0.42
JAPE	21.9	50.1	0.31	23.3	52.7	0.33
GCN-Align	17.7	37.8	0.25	19.3	41.5	0.27
SAEA w/o DB	38.8	69.2	0.50	42.4	70.3	0.52
SAEA	**44.1**	**70.7**	**0.53**	**44.5**	**71.0**	**0.53**

Table 3. Results on norm and cross-lingual datasets

Datasets	EN-FR			EN-DE		
Metrics	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR
MTransE	25.1	55.1	0.35	31.2	58.6	0.40
SEA	25.8	40.0	0.31	42.5	59.6	0.49
RSNs	34.7	63.1	0.44	48.7	72.0	0.57
IPTransE	25.5	55.7	0.36	31.3	59.2	0.41
BootEA	31.3	62.9	0.42	44.2	70.1	0.53
JAPE	25.6	56.2	0.36	32.0	59.9	0.41
GCN-Align	15.5	34.5	0.22	25.3	46.4	0.33
SAEA w/o DB	37.5	64.8	0.46	50.1	72.8	0.58
SAEA	**38.3**	**65.8**	**0.47**	**51.3**	**73.4**	**0.59**

– **GCN-Align**: A GCN-based approach for cross-lingual KG alignment. It generates node-level embeddings by encoding information about the nodes neighborhoods. Both the relation and attribute triples in KGs are employed to discover alignments.

Since other entity alignment models make use of more resources that are difficult to achieve, we omit comparisons against them.

5.4 Results

The evaluation results are presented in Table 2-5. SAEA w/o DB and SAEA represent our method. The latter one means that at the stage of random walk, degree-aware bias is implemented while the former one not. The best results are shown in bold among the group of methods. From the results, we have the following findings:

(1)Our proposed SAS consistently outperforms the state-of-the-art approaches on all datasets under different evaluation metrics. This observation verifies that SAEA can effectively model relational paths in KGs for

Table 4. Results on dense and monolingual datasets

Datasets	DBP-WD			DBP-YG		
Metrics	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR
MTransE	38.9	68.7	0.49	22.8	51.3	0.32
SEA	67.2	85.2	0.74	68.1	84.1	0.74
RSNs	76.3	92.4	0.83	82.6	95.8	0.87
IPTransE	43.5	74.5	0.54	23.6	51.3	0.33
BootEA	67.8	91.2	0.76	68.2	89.8	0.76
JAPE	39.3	70.5	0.50	26.8	57.3	0.37
GCN-Align	43.1	71.3	0.53	31.3	57.5	0.40
SAEA w/o DB	78.1	93.3	0.84	85.1	96.1	0.88
SAEA	**79.9**	**93.7**	**0.85**	**85.6**	**96.5**	**0.89**

Table 5. Results on dense and cross-lingual datasets

Datasets	EN-FR			EN-DE		
Metrics	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR
MTransE	37.7	70.0	0.49	34.7	62.0	0.44
SEA	62.3	85.7	0.71	65.2	79.4	0.70
RSNs	75.6	92.5	0.82	73.9	89.0	0.79
IPTransE	42.9	78.3	0.55	34.0	63.2	0.44
BootEA	64.8	91.9	0.74	66.5	87.1	0.73
JAPE	40.7	72.7	0.52	37.5	66.1	0.47
GCN-Align	37.3	70.9	0.49	32.1	55.2	0.40
SAEA w/o DB	80.2	93.7	0.84	75.7	89.3	0.81
SAEA	**80.6**	**94.2**	**0.85**	**77.6**	**91.2**	**0.82**

improving the accuracy of entity alignment and is capable of capturing long-term dependencies between entities effectively. In particular, compared with the second-best baseline RSN, our method achieves more than 3% improvement when matching the correct entity in the first position. This improvement is more obvious on monolingual datasets as the heterogeneity among them is more severe than one KG with different languages. In addition, it is worth mentioning that SAEA showed larger superiority over iterative and multi-resource-based approaches which manifests that our model fully explores the inherent structural features of KGs and can be enhanced by employing existing iteration strategies or combing extra resources. All the results show the advantage of our method.

(2)Our degree-aware random walk produces better entity representations for alignment. This comparison is drawn from the comparison of improvement made by SAEA w/o DB and SAEA. Especially on norm monolingual datasets, SAEA has more than 3% improvement on Hits@1 while on dense datasets, the improvement is slight. This indicates that degree-aware random walk plays a more important role in sampling paths in KGs where richer relational triples are scarce. Degree bias makes long-tail entities and relations more likely to be sampled so that the information between norm entities and long-tail

entities is balanced. How much this biased random walk can help is limited by the original entity distribution in KGs, but the improvement made from SAEA w/o DB to SAEA is justifiable.

6 Conclusions and Future Work

This paper presents a novel self-attentive heterogeneous sequence learning model for entity alignment. It discovers alignments based on learning embeddings from relational paths. In order not to ignore the unique triple structure of KGs, we design a crossed residual connection to focus on triples. Also, we introduce a degree-aware random walk for sampling paths in a balanced way. Our experiments on four real-world datasets demonstrated the effectiveness of our framework.

In future work, we plan to explore more advanced sequential models and graph neural networks for KG alignment task, such as MARINE [9] and MixHop [1]. Besides, how to combine extra resources in our approach is another interesting direction.

References

1. Abu-El-Haija, S., Perozzi, B., Kapoor, A., Harutyunyan, H., Alipourfard, N., Lerman, K., Steeg, G.V., Galstyan, A.: Mixhop: Higher-order graph convolution architectures via sparsified neighborhood mixing. arXiv preprint arXiv:1905.00067 (2019)
2. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of the 2008 ACM SIGMOD international conference on Management of data. pp. 1247–1250. ACM (2008)
3. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Advances in neural information processing systems. pp. 2787–2795 (2013)
4. Chen, M., Tian, Y., Chang, K.W., Skiena, S., Zaniolo, C.: Co-training embeddings of knowledge graphs and entity descriptions for cross-lingual entity alignment. arXiv preprint arXiv:1806.06478 (2018)
5. Chen, M., Tian, Y., Yang, M., Zaniolo, C.: Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. arXiv preprint arXiv:1611.03954 (2016)
6. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 855–864. ACM (2016)
7. Guo, L., Sun, Z., Hu, W.: Learning to exploit long-term relational dependencies in knowledge graphs. arXiv preprint arXiv:1905.04914 (2019)
8. Hu, W., Chen, J., Qu, Y.: A self-training approach for resolving object coreference on the semantic web. In: Proceedings of the 20th international conference on World wide web. pp. 87–96. ACM (2011)
9. Kawamae, N.: Marine: Multi-relational network embeddings with relational proximity and node attributes. In: The World Wide Web Conference. pp. 470–479. ACM (2019)

10. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., Van Kleef, P., Auer, S., et al.: Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. Semantic Web 6(2), 167–195 (2015)
11. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. pp. 3111–3119 (2013)
12. Ngomo, A.C.N., Auer, S.: 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)
13. Pei, S., Yu, L., Hoehndorf, R., Zhang, X.: Semi-supervised entity alignment via knowledge graph embedding with awareness of degree difference. In: The World Wide Web Conference. pp. 3130–3136. ACM (2019)
14. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 701–710. ACM (2014)
15. Raimond, Y., Sutton, C., Sandler, M.B.: Automatic interlinking of music datasets on the semantic web. LDOW 369 (2008)
16. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: Proceedings of the 16th international conference on World Wide Web. pp. 697–706. ACM (2007)
17. Sun, Z., Hu, W., Li, C.: Cross-lingual entity alignment via joint attribute-preserving embedding. In: International Semantic Web Conference. pp. 628–644. Springer (2017)
18. Sun, Z., Hu, W., Zhang, Q., Qu, Y.: Bootstrapping entity alignment with knowledge graph embedding. In: IJCAI. pp. 4396–4402 (2018)
19. Trisedya, B.D., Qi, J., Zhang, R.: Entity alignment between knowledge graphs using attribute embeddings. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 297–304 (2019)
20. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems. pp. 5998–6008 (2017)
21. Volz, J., Bizer, C., Gaedke, M., Kobilarov, G.: Discovering and maintaining links on the web of data. In: International Semantic Web Conference. pp. 650–665. Springer (2009)
22. Wang, Z., Lv, Q., Lan, X., Zhang, Y.: Cross-lingual knowledge graph alignment via graph convolutional networks. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. pp. 349–357 (2018)
23. Xu, K., Wu, L., Wang, Z., Feng, Y., Witbrock, M., Sheinin, V.: Graph2seq: Graph to sequence learning with attention-based neural networks. arXiv preprint arXiv:1804.00823 (2018)
24. Yang, Y., Sun, Y., Tang, J., Ma, B., Li, J.: Entity matching across heterogeneous sources. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1395–1404. ACM (2015)
25. Zhang, Q., Sun, Z., Hu, W., Chen, M., Guo, L., Qu, Y.: Multi-view knowledge graph embedding for entity alignment. arXiv preprint arXiv:1906.02390 (2019)
26. Zhu, H., Xie, R., Liu, Z., Sun, M.: Iterative entity alignment via joint knowledge embeddings. In: IJCAI. pp. 4258–4264 (2017)