# Self-learning and Embedding Based Entity Alignment

Saiping Guan, Xiaolong Jin, Yantao Jia, Yuanzhuo Wang, Huawei Shen and Xueqi Cheng
CAS Key Laboratory of Network Data Science and Technology,
Institute of Computing Technology, Chinese Academy of Science
School of Computer and Control Engineering, University of Chinese Academy of Sciences
guansaiping@software.ict.ac.cn; {jinxiaolong, jiayantao, wangyuanzhuo, shenhuawei, cxq}@ict.ac.cn

***Abstract*—Entity alignment aims to identify semantical matchings between entities from different groups. Traditional methods (e.g., attribute comparison based methods, clustering based methods, and active learning methods) are usually supervised by labelled data as prior knowledge. Since it is not trivial to label data for training, researchers have recently turned to unsupervised methods, and have thus developed similarity based methods, probabilistic methods, hierarchical graph model based methods, etc. As an important part of a knowledge graph, entities contain rich semantical information that can be well learned by knowledge graph embedding in a low-dimensional vector space. However, existing methods for entity alignment have paid little attention to knowledge graph embedding. In this paper, we propose a Self-learning and Embedding based method for Entity Alignment, thus called SEEA, to iteratively find semantically matched entity pairs, which makes full use of semantical information contained in the attributes of entities. Experiments on two realistic datasets and comparison with the baselines validate the effectiveness and merits of the proposed method.**

***Index Terms*—entity alignment; self-learning; knowledge graph embedding.**

## I. Introduction

The rapid development of Internet technology and Web applications triggers the explosion of various open-accessed data, which contains a large amount of valuable knowledge. Knowledge graph was thus developed to organize these valuable knowledge in a systematical and semantical manner. Entity alignment, also known as entity matching and record linkage, plays an important role in knowledge graph construction, as well as the integration of existing knowledge graphs. Entity alignment aims to align entities between different entity groups or even different knowledge graphs. It has been extensively studied in recent years. According to whether or not using labeled data, the methods for entity alignment can be classified into supervised and unsupervised ones.

Supervised methods are usually based on attribute comparison [1–4], clustering [5], active learning [6, 7], etc. They learn models from labelled data to conduct entity alignment. The major problem of supervised methods is that it is arduous and time-consuming to label data. Unsupervised methods include similarity based ones [8, 9], probabilistic ones [10], hierarchical graph model based ones [11], etc. These methods usually need to define efficient measures of similarity or consider probability based on some assumptions. For example, in [9], entity alignment is formulated as a classifier that relies on a complex similarity function consisting of a combination of atomic similarity measures. In [10], probabilities based on logical rules are used to determine whether two entities are aligned or not. In order to transform the equations derived from logical rules into probability assessments, the authors assume mutual independence of all distinct elements of the models, which is of course not true in practice, as they stated.

Recently, knowledge graph embedding has been successfully applied to various knowledge graph related research issues [12–17]. A few studies have also employed knowledge graph embedding to conduct entity alignment, which do not need to define similarity measures or make assumptions in advance. Actually, as an important part of a knowledge graph, entities contain rich semantical information, which can be utilized to conduct or promote the performance of entity alignment. Hao et al. [18] proposed a method that jointly learns the embeddings of multiple knowledge graphs in a uniform vector space to align entities. This is the first study to deal with the alignment problem using an end-to-end joint embedding method. More recently, Cai et al. [19] proposed cross-KG to simultaneously learn embeddings for two different knowledge graphs. Their method achieves good performance in terms of Hits@1 (the proportion of correctly aligned entities ranked in top1) in entity alignment. However, they treated all relations equally, even though some relations are actually the attributes of entities. In this sense, these existing knowledge graph embedding based methods for entity alignment are not expressive enough, as they do not differentiate the attributes of entities from the relations between entities.

In this paper, we propose an unsupervised method, called Self-learning and Embedding based Entity Alignment (SEEA), which embeds entities, attributes, and attribute values into a low-dimensional vector space, while clearly differentiating relations and attributes, and conducts entity alignment in an iterative self-learning manner. Here, by self-learning we mean that SEEA adopts entity alignment pairs learned from the previous iteration to update the embeddings of entities, attributes, and attribute values in the next iteration, and in turn, the embeddings adjust the entity alignment results.

In general, the main contributions of this paper is three-fold:

- It presents a more expressive knowledge graph embedding that can well differentiate the attributes of entities, which are weighed differently according to their importance, and the relations between entities;

- It proposes an iterative self-learning mechanism that utilizes the alignment results in the previous iteration to update the embeddings of entities, attributes, and attribute values in the next iteration;
- The proposed method are evaluated on two realistic datasets of different domains and in different languages. Experimental results demonstrate that the proposed method outperforms the baseline methods and possesses good adaptability regardless of domains.

The rest of this paper is organized as follows. We first present a review of related works in Section II. Then, in Section III, we formulate the problem. In section IV, we describe in detail the proposed method and demonstrate the experimental results in the following section. Finally, the paper is concluded in Section VI.

## II. Related Works

In this section, we first review the related works on entity alignment and knowledge graph embedding.

### A. Entity Alignment

*1) Common Methods:* As mentioned above, the existing methods for entity alignment can basically be categorized into supervised and unsupervised ones.

*a) Supervised Methods:* Supervised methods compare attributes, cluster entities or conduct active learning. Attribute comparison based methods usually employ traditional machine learning techniques, such as, decision tree [1], Support Vector Machine (SVM) [2], and ensemble learning [3], to determine whether entity pairs align or not. Lacoste et al. [4] proposed a simple greedy matching algorithm, named SiGMa, to align knowledge bases. SiGMa starts with a small number of seed alignment pairs of good quality, then it incrementally augments the alignment pairs according to the string, properties and structures information in a greedy local search manner. Cohen and Richman [5] presented an entity-name clustering based method for realizing entity alignment in a scalable and adaptive way. Sarawagi and Bhamidipaty [6] designed an active learning based entity alignment system, called ALIAS, which allows user to label challenging entity pairs, then it uses the label feedback together with the initial entity alignment training data to train a classifier. Furthermore, Arasu et al. [7] integrated active learning and blocking scheme to reduce the number of labeling requests. The algorithm picks examples that are most informative for training entity alignment classifier to be labeled.

*b) Unsupervised Methods:* Labeling prior knowledge to train supervised methods takes a lot of time and is formidable. Therefore, unsupervised methods are more preferred in the community. Unsupervised methods can be further divided into similarity based, probabilistic or hierarchical graph model based ones. For instance, Nikolov et al. [8] established decision rules in an unsupervised way to align entities, relying on the characteristics of entities and the similarity distributions between entities. Ngomo and Lyko [9] proposed an efficient and unsupervised entity alignment method, called EUCLID, which employs different similarity measures to evaluate the properties of the entities and get entity alignment pairs, then depending on different heuristics, it updates the alignment until the max number of iterations is reached. Suchanek et al. [10] proposed an unsupervised probabilistic method, PARIS, to conduct entity alignment. In PARIS, it first claims the logic rules that entity alignment pairs should satisfy, then these logic rules are formulated as equalities, and transformed into probability assessments to evaluate the probability whether two entities can be aligned. Ravikumar and Cohen [11] described a hierarchical graphical model framework for the entity alignment problem. It generalizes a two-layer generative model (including feature vector layer and match layer) to a three-layer hierarchical graphical model by adding an intermediate latent variables layer. Each node in the intermediate layer corresponds to the distance feature of a property pair. And the top match layer depends on the latent variables of these individual properties.

*2) Methods Based on Structure or Class:* In the above direct entity alignment, two entities are considered to be aligned when their similarity exceeds a threshold. It is not suitable, when the overlap between the datasets is small, as [20] states. Therefore, Jiménez-Ruiz and Grau [21] proposed a highly scalable alignment system, LogMap. It first lexically and structurally indexes input ontologies, i.e., it indexes the labels of the classes in each ontology as well as their lexical variations and expansion, and it represents the extended class hierarchy of each input ontology. Then it uses the lexical indexes to derive an initial set of alignment pairs, almost exactly lexical correspondences. After that, it alternates alignment repair and alignment discovery steps according to structurally indexes in an iterative process to get final almost clean alignment results. In [20], a class based alignment method, SERIMI, is proposed, which combines direct alignment with a novel class based alignment technique. A class is a set of entities where each entity must share at least one feature with any other entity in this set. Specifically, given a class of entities from the source dataset, and a set of candidate alignments retrieved from the target dataset, SERIMI refines the candidate alignments by filtering out those that do not share some features with others and gets a class of matches. Then, it performs direct alignment to get correct entity alignment pairs. Ngo and Bellahsene [22] proposed YAM++, one of the best system in OAEI (Ontology Alignment Evaluation Initiative) competitions from 2011 to 2013. It consists of element level and structure level. In the element level, YAM++ discovers as many as possible high accuracy alignment pairs by analyzing elements independently, ignoring their relations with others. The alignment pairs are then passed to the structure level in order to discover new alignment pairs by analyzing the position of the entities.

### B. Knowledge Graph Embedding

In the real-world scenarios, a multitude of knowledge bases denote the same entities in different ways, because they are created by different individuals or organizations with different management mechanisms. As an example, a movie in the Baidu movie website[1] has its unique identifier, while the identifier of the aligned movie in the Douban movie website[2] is different. Since direct content information overlap about the two movie entities themsevlves is unavailable (they are just different identifiers), methods involving direct content information are invalid. One possible solution is to turn to their attribute information completely, and learn latent-embeddings of entities based on this structure to conduct entity alignment. As an important part of a knowledge graph, entities can get semantical embeddings through knowledge graph embedding methods.

TransE [12] opens a line of translation based methods, which learn the embeddings of entities and relations to support further applications, such as link prediction, triples classification, etc. TransE interprets relations as translations operating on low-dimensional embeddings of the entities. If a triple $(h, r, t)$ holds, then the embedding (denoted with the same letter in boldface) of the tail entity $t$ should be close to that of the head entity $h$ plus a vector that depends on the relationship $r$, otherwise, far away. The score function has a form of $\|\mathbf{h}+\mathbf{r}-\mathbf{t}\|_{L_1/L_2}$. TransE is simple and effective. Following the translation based line, a multitude of methods for dealing with more complicated relations spring up, such as TransH [13], (C)TransR [14], TransA [23], etc.

However, these method ignore the heterogeneity (some relations link many entity pairs and others do not) and the imbalance (the number of head entities and that of tail entities in a relation could be different) of knowledge graphs. Therefore, Ji et al. [15] proposed TranSparse to deal with these two issues. In TranSparse, transfer matrices are replaced by adaptive sparse matrices, whose sparse degrees are determined by the number of entities (or entity pairs) linked by relations. In many knowledge graphs, some relations indicate attributes of entities and others indicate relations between entities. To distinguish between relations and attributes, Lin et al. [16] proposed KR-EAR. In KR-EAR, one-to-many and many-to-one relations are viewed as attributes, and they are learned differently. The above methods are too strict to model the complex and diverse entities and relations. Thus, Feng et al. [17] proposed Flexible Translation (FT), which regards relations as translation between the head entity vector and the tail entity vector with flexible magnitude. In FT, the principle is to enforce that $\mathbf{h}+\mathbf{r}$ has the same direction with $\mathbf{t}$ and $\mathbf{t}-\mathbf{r}$ has the same direction with $\mathbf{h}$, instead of strictly constraining $\mathbf{h}+\mathbf{r}=\mathbf{t}$.

Recent common methods and structure or class based methods in entity alignment usually ignore the latent entity embeddings from these knowledge graph embedding methods, which can be used to perform sematical entity alignment. Hao et al. [18] first focused on embedding based method to align entities. Then, Cai et al. [19] recently proposed cross-

[1]http://movie.baidu.com/
[2]https://movie.douban.com/

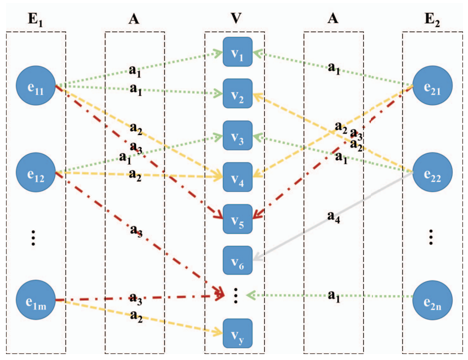

Fig. 1: An illustrative example of the knowledge graph $G$.

KG, which embeds different KGs into a uniform semantic space simultaneously, and use the learned entity embeddings to conduct entity alignment. Nonetheless, they regard all relations equally. This paper, in contrast, aims at performing more semantical entity alignment by knowledge graph embedding differentiating relations and attributes. Furthermore, a self-learning technique is used to label data automatically to retrain the method and make full use of the data. The idea somehow likes the self-tuning strategy in TRA [24]. TRA is a distributed computing framework for assembling taxonomy matchers and generate optimal taxonomy mappings by threshold rank aggregation algorithm with a self-tuning based strategy. However, TRA tunes the threshold parameter and we adjust the alignment results.

## III. Problem Formulation

Let $G = (E, A, V, R, AT, RT)$ be a knowledge graph, where $E = E_1 \cup E_2$ is the entity set, $E_1$ and $E_2$ are the two groups of entities to be aligned; $A$, $V$, and $R$ denote the attribute set, the attribute value set, and the relation set respectively; $AT \subseteq E \times A \times V$ is the set of attributional triples and $RT \subseteq E_1 \times R \times E_2$ is the set of relational triples between the two entity groups $E_1$ and $E_2$. In this paper, alignment relation $r$ is the only type of relation, i.e., $R = \{r\}$.

Figure 1 presents an illustrative example of the knowledge graph $G$. Here, $m$, $n$, $x$ and $y$ are the size of $E_1$, $E_2$, $A$ and $V$ respectively. The two groups of entities to be aligned are $E_1 = \{e_{11}, e_{12}, \cdots, e_{1m}\}$ and $E_2 = \{e_{21}, e_{22}, \cdots, e_{2n}\}$ respectively. Note that $m$ is not necessarily equal to $n$. The total attribute set is $A = \{a_1, a_2, \cdots, a_x\}$, and the total attribute value set is $V = \{v_1, v_2, \cdots, v_y\}$. Each entity is denoted as a labeled circle, each attribute is denoted as a labeled directed edge with different line styles and different colors, and each attribute value is denoted as a labeled rectangle. In this example, $e_{11}$ in the first group has two attribute values $v_1$ and $v_2$ for attribute $a_1$. Take the movie dataset for example, it is a common phenomenon that a movie has many actors. And the actor of a movie can also be the director of another movie. It is just the case if $a_1$ stands for actor and $a_2$ stands for director, then $v_2$ is an actor of the movie entity $e_{11}$ in $E_1$ and also a director of $e_{22}$ in $E_2$. All these attributional triples

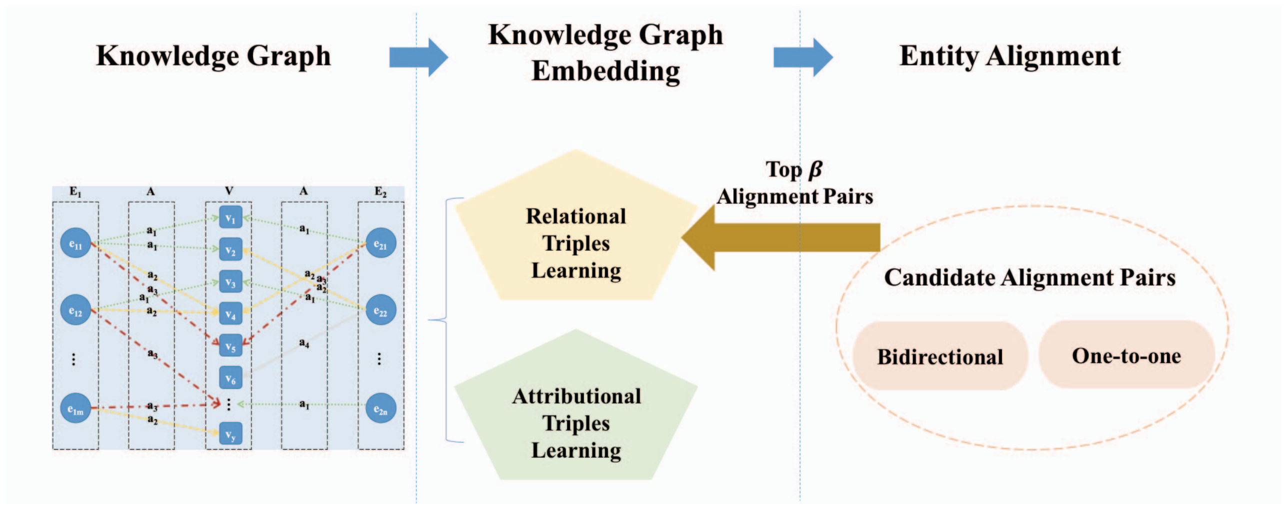
Knowledge Graph
Knowledge Graph Embedding
Entity Alignment
Top β Alignment Pairs
Relational Triples Learning
Attributional Triples Learning
Candidate Alignment Pairs
Bidirectional
One-to-one

Fig. 2: The framework of the proposed SEEA method.

form the attributional triple set $AT$. And at the beginning, the relational triple set $RT$ is empty. After each iteration, some relational triples are added in, that is, $(e_{11}, r, e_{21})$ may be added in.

For such a knowledge graph $G$, the objective is to learn the embeddings $\mathbf{X}$ of entities, attributes, and attribute values in order to find alignment pairs between entities in $E_1$ and $E_2$. The multi-value characteristic of attribute values described above makes it difficult to learn embeddings. The embeddings are in a vector space $\mathbb{R}^k$, where $k$ is the latent dimension of the embedding space. In this paper, we denote the embeddings with the same letters in boldface.

## IV. The SEEA Method

The SEEA method takes knowledge graph as input and consists of two fundamental components: knowledge graph embedding and entity alignment. Specifically, a self-learning mechanism serves as the feedback operation from entity alignment to knowledge graph embedding. Figure 2 illustrates the whole framework of the proposed SEEA method. Next, we will introduce the method in detail.

### A. Relational Triples Learning

For a relational triple $(h, r, t)$, we optimize the conditional probability $P(h|r, t, \mathbf{X})$, $P(t|h, r, \mathbf{X})$ and ignore $P(r|h, t, \mathbf{X})$, since in this paper, the relation $r$ stands for only one relation, i.e., the alignment relation. $P(h|r, t, \mathbf{X})$ and $P(t|h, r, \mathbf{X})$ are formalized in the same way following the negative sampling used in Word2Vec [25]. Take $P(h|r, t, \mathbf{X})$ for example, it is formalized as follows:

$$P(h|r,t,\mathbf{X}) = \prod_{(h,r,t)\in RT} \Bigg[\sigma\big(g(\mathbf{h},\mathbf{r},\mathbf{t})\big) \prod_{i=1}^{c_1'} \mathbb{E}_{(h_i,r,t)\sim P(RT^-)}\sigma\big(g(\mathbf{h_i},\mathbf{r},\mathbf{t})\big)\Bigg], \tag{1}$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function, $RT^- = \{(h', r, t)\}$ is the set of negative triples, composed of training relational triples with the head entities replaced by random entities from the same group, $\mathbb{E}_{(h_i,r,t)\sim P(RT^-)}$ is the event that randomly samples instances from $RT^-$, $c_1'$ is the number of negative samples for each valid relational triple (with the head entity replaced), and $g(\mathbf{h}, \mathbf{r}, \mathbf{t})$ is the score function. We hope $\mathbf{h}$ lies close to $\mathbf{t}$ when $h$ and $t$ can be aligned; otherwise, $\mathbf{h}$ should be far away from $\mathbf{t}$. Hence, we calculate the score function of the relational triple as follows:

$$g(\mathbf{h},\mathbf{r},\mathbf{t}) = -\|\mathbf{h}-\mathbf{t}\|_{L_1}. \tag{2}$$

In experiments, for each valid relational triple $(h, r, t)$, we sample a number, $c_1$, of negative triples by randomly replacing head entity $h$ or tail entity $t$ with other entities from the same group. That is, we denote $c_1'$ plus the number of negative samples with the tail entities replaced for $P(t|h, r, \mathbf{X})$ as $c_1$.

### B. Attributional Triples Learning

For an attributional triple $(h, a, v)$, following pTransE (Probabilistic TransE) [26], we formulate the likelihood $L\big((h, a, v)|\mathbf{X}\big)$ as follows:

$$\begin{aligned} L\big((h,a,v)|\mathbf{X}\big) = &\log P(h|a,v,\mathbf{X}) + \log P(a|h,v,\mathbf{X}) \\ &+ \log P(v|h,a,\mathbf{X}). \end{aligned} \tag{3}$$

However, attributes are the primary source of one-to-many and many-to-one relations [16]. Intuitively, given the entity and attribute, the attribute values is determined. While, given the attribute and attribute value, the entity is somehow uncertain. And the correlations between entities and their attributes should be captured with a classification model [16]. Therefore, for each attributional triple, we just consider $\log P(v|h, a, \mathbf{X})$

and optimize the conditional probability $P(v|h,a,\mathbf{X})$. Following equation (1), we compute $P(v|h,a,\mathbf{X})$ as follows:

$$P(v|h,a,\mathbf{X}) = \prod_{(h,a,v)\in AT} \Bigg[\sigma\big(\varphi(\mathbf{h},\mathbf{a},\mathbf{v})\big) \prod_{i=1}^{c_2} \mathbb{E}_{(h,a,v_i)\sim P(AT^-)}\sigma\big(\varphi(\mathbf{h},\mathbf{a},\mathbf{v_i})\big)\Bigg], \tag{4}$$

where $AT^- = \{(h,a,v')\}$ is the set of negative triples, composed of training attributional triples with the attribute values replaced by random attribute values and $c_2$ is the number of negative samples for each valid attributional triple. Note that, for a particular attributional triple $(h,a,v)$, its corrupted triples may also exist in the knowledge graph $G$ that should be regarded as valid triples. In this case, they should be removed (namely, filtered out) before getting the performance of the method. Here, we follow KR-EAR [16] to define the score function $\varphi(\mathbf{h},\mathbf{a},\mathbf{v})$ as follows:

$$\varphi(\mathbf{h},\mathbf{a},\mathbf{v}) = -\|f(\mathbf{h}\mathbf{W_a}+\mathbf{a})-\mathbf{v}\|_{L_1}+b, \tag{5}$$

where $f()$ is a nonlinear function, $\tanh$ is used in this paper and $b$ is a bias constant. It projects an entity into the attribute space via a single-layer neural network, then calculates the semantic similarity between its projected embedding and the embedding of its corresponding attribute value.

Moreover, we consider the importance of different attributes by assigning them with different weights. As an example, the name of a movie is more important than its other attributes and thus should be given a higher weight.

### C. Self-learning Mechanism

As described previously, the proposed method operates in an iterative manner in order to adopt knowledge learned from one iteration to update the embeddings of entities, attributes and attribute values in the next iteration. Specifically, in self-learning, we use the learned relational triples to update all the embeddings in the next iteration. In this paper, we determine the relational triples in terms of the cosine similarity of learned entity embeddings. Note that the relational triples should be bidirectional, namely, the head entity chooses the tail entity at the first place, and vice versa. We then adopt the top $\beta$ relational triples as the prior knowledge to retrain the method. This progress is carried out iteratively until it converges. If no new relational triple can be learned in the top $\beta$ relational triples, that is, all the top $\beta$ relational triples were added in $RT$ in previous iterations, the algorithm converges.

Next, we give a short proof of the convergence above. Firstly, the self-learning is a monotonically increasing process. In each iteration, at least one new relational triple is added to the relational triple set $RT$ and the total number of added relational triples monotonically increases. Secondly, this process has its upper bound, the total number of added relational triples is less than or equal to $\min(m,n)$ ($m$ and $n$ are the size of $E_1$, $E_2$ respectively, as stated in section III). Therefore, the self-learning process can converge after a certain number of iterations. In experiments, we find that the self-learning process converges only after several iterations.

### D. Method Overview and Objective Function

Overall, the whole SEEA method is described in Algorithm 1.

**Algorithm 1** The self-learning based entity alignment algorithm.

```
Input: the attributional triple set $AT$ from two groups, $\beta$;
Output: the final alignment pairs;
 1: for $i = 1$ to $\infty$ do
 2:   if $i = 1$ then
 3:     Initialize the embeddings randomly. Model $(h,a,v) \in AT$ to get the embeddings of entities, attributes and attribute values;
 4:   else
 5:     Initialize the embeddings with the results from the $(i-1)$ iteration. Model $(h,a,v) \in AT$ and $(h,r,t) \in RT$ to update all the embeddings;
 6:   end if
 7:   For each entity in the first group, calculate the most similar entity in the second group according to cosine similarity and store the pair with their similarity in $p_1$;
 8:   For each entity in the second group, calculate the most similar entity in the first group according to cosine similarity and store the pair with their similarity in $p_2$;
 9:   if no new alignment pair exists in the top $\beta$ of $p_1 \cap p_2$ then
10:     Return the $p_1 \cap p_2$ in current iteration;
11:     break;
12:   else
13:     Construct new relational triples according to new alignment pairs in the top $\beta$ of $p_1 \cap p_2$ and add them to the relational triple set $RT$;
14:   end if
15: end for
```

Finally, we adopt stochastic gradient descent to minimize the objective function:

$$O(\mathbf{X}) = \prod_{(h,r,t)\in RT} P\big((h,r,t)|\mathbf{X}\big) \prod_{(h,a,v)\in AT} P(v|h,a,\mathbf{X}) + \lambda C(\mathbf{X}), \tag{6}$$

where $\lambda$ is a hyper-parameter for weighting the regularization factor $C(\mathbf{X})$, which contains the regularizations of the embeddings of entities, attributes and attribute values. In the process of vector regularization, we use the widely used $L_2$-norm. It is defined as follows:

$$C(\mathbf{X}) = \sum_{h\in E}\big[\|\mathbf{h}\|_{L_2}-1\big]_+ + \sum_{v\in V}\big[\|\mathbf{v}\|_{L_2}-1\big]_+ + \sum_{h\in E}\sum_{i=1}^{\|A\|}\big[\|f(\mathbf{h}\mathbf{W_i}+\mathbf{A_i})\|_{L_2}-1\big]_+ , \tag{7}$$

TABLE I: Complexities of several embedding methods (in one epoch).

| Method | Parameter Complexity | Time Complexity |
|---|---|---|
| TransE | $O\big((N_E + y - N_{E\cap V} + x)k\big)$ | $O(N_{AT}k)$ |
| TranSparse | $O\big((N_E + y - N_{E\cap V})k + 2x(1-\theta)(k+1)k\big)$ $(0 \ll \theta \le 1)$ | $O\big(2(1-\theta)N_{AT}k^2\big)$ $(0 \ll \theta \le 1)$ |
| TransE-FT | $O\big((N_E + y - N_{E\cap V} + x)k\big)$ | $O(N_{AT}k)$ |
| cross-KG | $O\big((N_E + y - N_{E\cap V} + x)k\big)$ | $O\big((N_{AT} + N_{MP})k\big)$ |
| KR-EAR | $O\big((N_E + y + x)k + xk^2\big)$ | $O(N_{AT}k^2)$ |
| SEEA | $O\big((N_E + y - N_{E\cap V} + x)k + xk^2\big)$ | $O(N_{AT}k^2)$ $(i = 1)$ <br> $O(N_{AT}k^2 + N_{RT}k)$ $(i > 1)$ |

where $\mathbf{W_i}$ and $\mathbf{A_i}$ are the matrix and the attribute embedding of the $i$-th attribute, respectively.

### *E. Complexity Analysis*

As shown in Table I, we compare parameter complexity and time complexity of various methods with our method in terms of code implementation. We denote $N_E$, $N_{AT}$, $N_{RT}$ as the size of the entity set $E$, the attributional triple set $AT$ and the relational triple set $RT$ respectively. $N_{E\cap V}$ is the size of the intersection of entity and attribute value sets. $\theta$ denotes the average sparse degree of all sparse transfer matrices. And the sparse degrees are determined by the number of head (tail) entities linked by relations. Cross-KG needs an initial alignment set $MP$, thus we treat the attributes and attribute values which appear in both of the two groups of data to be aligned initially and add them to $MP$. The size of $MP$ is denoted as $N_{MP}$. Recall that $x$, $y$ and $k$ are the size of the attribute set $A$, the attribute value set $V$, and the latent dimension of the embedding space, respectively.

*1) Parameter Complexity:* Compared to KR-EAR that also distinguishes between relations and attributes, the proposed SEEA method reduces parameter complexity. In the code implementation of KR-EAR, if an entity is the attribute value of some other entity and itself has its attributes and values, then this entity has two embedding representations: one for entity itself and the other for the attribute value. As an example, "Samantha Brown" is the actor of the movie "The Present" and "Samantha Brown" has her own attributes, such as type, the according attribute value is person. In this case, "Samantha Brown" has two embeddings for entity and value respectively. While in SEEA, only one vector is used for each entity in $E \cup V$. Each entity maps to a vector, regardless of the roles it plays in the triples. Since the embedding represents the entity, it can record the information of the entity, including different attributes information from different training triples. Therefore, SEEA can reduce $N_{E\cap V}$ parameter complexity. As $N_{E\cap V}$ may be very large in real applications, it can reduce a large number of parameters.

*2) Time Complexity:* Table I demonstrates the time complexity in one epoch. Note that SEEA is a self-learning method and it contains several iterations, each with many epoches. However, other methods only contain one iteration with many epoches. Here we state that the self-learning process does not increase much time complexity. As the self-learning process is used to update the embeddings and alignment results from the last iteration, after the first iteration, in the following iteration, the number of the embedding training epoch can be much smaller (at most 100 in this paper compared to 500 in the first iteration). Based on the embeddings from the last iteration, SEEA moves positions of the existed embedding vectors according to new relational triples. The process is fast with the help of embeddings from the last iteration. As the number of iterations is small and after the first iteration, each following iteration does not take much time, the time complexity does not increase much compared to KR-EAR.

## V. Experiments

In this section, we will introduce the datasets and experiments in detail.

### *A. Datasets*

The experiments were carried out on two datasets of different domains and in different languages. Each dataset contains two groups of entities and their attribute information for alignment. The first dataset is derived from Cora[3] in English, denoted as Cora1. Cora includes bibliographical information about scientific papers, extracted from citations in scientific papers. Each entity contains the following attributes: author, title, publisher, etc. We pick two entities randomly from each duplicate set in Cora and assign one to the first group and the other to the second group in Cora1. The unique entities in Cora are assigned to one of the two groups randomly, which can be regarded as noise entities that can not find the aligned entities in the other group. And we keep the corresponding attribute information of the assigned entities in the attributional triple set $AT$. The second one is Baidu and Douban Movie/TV Shows information dataset in Chinese, denoted as Baidu_Douban_M/TV. It is extracted from two well-known websites, Baidu and Douban respectively, and each forms one of the two groups to be aligned. Each entity contains the following attributes: name, actor, director, type and release time. The statistics of the datasets is shown in Table II. Here, $N_{E\cup V}$ is the size of the union of the entity set $E$ and the attribute value set $V$, and $N_P$ is the number of ground truth entity alignment pairs.

TABLE II: The statistics of the two datasets.

| Dataset | $N_{E\cup V}$ | $x$ | $N_{AT}$ | $m$ | $n$ | $N_P$ |
|---|---|---|---|---|---|---|
| Cora1 | 1441 | 16 | 2180 | 145 | 143 | 116 |
| Baidu_Douban_M/TV | 8143 | 5 | 27960 | 762 | 762 | 762 |

[3]https://hpi.de/naumann/projects/data-quality-and-cleansing/dude-duplicate-detection.html#c115302

TABLE III: Experimental results on entity alignment (%).

| Dataset | Cora1 | | | Baidu_Douban_M/TV | | |
|---|---|---|---|---|---|---|
| Metric | Precision | Recall | F1-measure | Precision | Recall | F1-measure |
| TransE | 87.06 | 63.79 | 73.63 | 98.97 | 88.58 | 93.49 |
| TranSparse | 89.28 | 64.66 | 75.00 | 99.41 | 88.06 | 93.39 |
| TransE-FT | 88.58 | 68.10 | 77.00 | 99.51 | 89.71 | 94.35 |
| cross-KG | 92.04 | 65.69 | 76.64 | 99.31 | 82.76 | 90.28 |
| KR-EAR | 84.35 | 69.66 | 76.30 | 99.09 | 85.79 | 91.96 |
| SEEA(without sl) | 90.72 | 75.86 | 82.63 | 99.61 | 90.23 | 94.69 |
| SEEA | **93.75** | **77.59** | **84.90** | **99.63** | **90.72** | **94.96** |

The size of the intersection of the entity set and the attribute value set, denoted by $N_{E \cap V}$, of Baidu_Douban_M/TV dataset is 5695, which is quite large compared to that of the value set, 6619. In this case, the proposed SEEA method reduces a lot of parameter complexity compared to KR-EAR.

## B. Baselines and Metrics

In the scenario addressed in this study, the entities from two groups have unique identifiers and share nothing in entities themselves. The commonly used method for entity alignment, such as LogMap [21] and YAM++ [22], which need common entity information between the two groups, are not suitable for the addressed scenario. Therefore, we adopt six latent-embedding based methods, i.e., TransE, TranSparse, TransE-FT, cross-KG and KR-EAR, which are either simple, effective or state-of-the-art, as well as SEEA without self-learning, denoted as SEEA(without sl) as baselines. In the SEEA and SEEA(without sl) experiments, we set higher weights to important attributes, e.g., paper title in Cora1 and movie name in Baidu_Douban_M/TV, denoted as $w$. The parameters of the methods are tuned in the full grid. The optimal settings are: $k = 80$, $c_1 = 10$, $c_2 = 1$, $b = 7$, $\lambda = 0.9$, $\beta = 6$, $w = 8$ for Cora1 and $k = 100$, $c_1 = 10$, $c_2 = 1$, $b = 7$, $\lambda = 0.7$, $\beta = 9$, $w = 2$ for Baidu_Douban_M/TV. Precision, Recall and F1-measure are used as metrics for the purpose of comparison.

## C. Experimental Results

Table III presents the results of all methods on the two datasets. It can be seen from Table III that on both datasets, SEEA performs better than all the baselines, which demonstrates the superiority of the proposed method in different domains. In more detail, in Cora1, SEEA(without sl) achieves higher Recall and F1-measure than other five baselines. SEEA further increases all metrics. It specifically increases Recall and F1-measure by around 8%, as compared to the five baselines. In Baidu_Douban_M/TV, SEEA also demonstrates the best performance in terms of all metrics. This is because SEEA takes rich information contained in entities, including attributes, attribute values and their interactions, into consideration, and gives higher weights to important attributes. Furthermore, SEEA takes full advantage of the data in an iteratively self-learning manner. The relational triples from the last iteration is used to adjust all the embeddings, which in turn enhances the alignment results.

The performance of SEEA in Baidu_Douban_M/TV is better than that in Cora1 (e.g., SEEA obtains 90.72% Recall in Baidu_Douban_M/TV, while obtaining 77.59% Recall in Cora1), the reason for this may be two-fold: On the one hand, the size of the attribute set in Cora1 is larger (16 in Cora1 compared to 5 in Baidu_Douban_M/TV), the data sparsity for attributes is more serious, which makes the alignment task more formidable; On the other hand, English text is more diversified. Take English name for example, in some citations, the first name is placed before the last name, but in some other citations, the first name follows the last name. Moreover, various abbreviations are widely used.

## VI. Conclusions

In this paper, we proposed a self-learning and embedding based unsupervised method, called SEEA, for aligning entities in different groups. SEEA clearly differentiates the attributes of entities and the relations between entities when learning the embeddings, and assigns higher weights to important attributes. Note that the alignment relation between entities from different groups is the only relation type in this study. At the beginning, there is no relational triple, i.e., no any aligned entity pair. But, relational triples are gradually learned from entity groups by an iterative self-learning mechanism, which specifically utilizes the alignment results in the previous iteration to update the embeddings of entities, attributes, and attribute values in the next iteration. This process iterates until it converges, i.e., no new relational triple is learned. Experimental comparisons with the baselines demonstrate that the proposed SEEA outperforms the baselines in two realistic datasets of different domains and in different languages, manifesting the superiority and merits of SEEA.

## Acknowledgment

This work is supported by National Key Research and Development Program of China under grant 2016YFB1000902, National Grand Fundamental Research 973 Program of China under grant 2014CB340401, and National Natural Science Foundation of China under grants 61572473, 61572469, 61402442, 91646120, and 61402022.

## References

[1] M. G. Elfeky, V. S. Verykios, and A. K. Elmagarmid, "Tailor: A record linkage toolbox," in *Proceedings of the 18th International Conference on Data Engineering (ICDE'02)*, 2002, pp. 17–28.

[2] M. Bilenko and R. J. Mooney, "Adaptive duplicate detection using learnable string similarity measures,"

in *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'03)*, 2003, pp. 39–48.

[3] Z. Chen, D. V. Kalashnikov, and S. Mehrotra, "Exploiting context analysis for combining multiple entity resolution systems," in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data (SIGMOD'09)*, 2009, pp. 207–218.

[4] S. Lacoste-Julien, K. Palla, A. Davies, G. Kasneci, T. Graepel, and Z. Ghahramani, "Sigma: Simple greedy matching for aligning large knowledge bases," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'13)*, 2013, pp. 572–580.

[5] W. W. Cohen and J. Richman, "Learning to match and cluster large high-dimensional data sets for data integration," in *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'02)*, 2002, pp. 475–480.

[6] S. Sarawagi and A. Bhamidipaty, "Interactive deduplication using active learning," in *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'02)*, 2002, pp. 269–278.

[7] A. Arasu, M. Götz, and R. Kaushik, "On active learning of record matching packages," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data (SIGMOD'10)*, 2010, pp. 783–794.

[8] A. Nikolov, M. dAquin, and E. Motta, "Unsupervised learning of link discovery configuration," in *Proceedings of the 9th International Conference on The Semantic Web: Research and Applications (ESWC'12)*, 2012, pp. 119–133.

[9] A.-C. N. Ngomo and K. Lyko, "Unsupervised learning of link specifications: Deterministic vs. non-deterministic," in *Proceedings of the 8th International Conference on Ontology Matching - Volume 1111 (OM'13)*, 2013, pp. 25–36.

[10] F. M. Suchanek, S. Abiteboul, and P. Senellart, "Paris: Probabilistic alignment of relations, instances, and schema," *Proceedings of the VLDB Endowment*, vol. 5, no. 3, pp. 157–168, 2011.

[11] P. Ravikumar and W. W. Cohen, "A hierarchical graphical model for record linkage," in *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence (UAI'04)*, 2004, pp. 454–461.

[12] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Proceedings of the 26th International Conference on Neural Information Processing Systems (NIPS'13)*, 2013, pp. 2787–2795.

[13] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI'14)*, 2014, pp. 1112–1119.

[14] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI'15)*, 2015, pp. 2181–2187.

[15] G. Ji, K. Liu, S. He, and J. Zhao, "Knowledge graph completion with adaptive sparse transfer matrix," in *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI'16)*, 2016, pp. 985–991.

[16] Y. Lin, Z. Liu, and M. Sun, "Knowledge representation learning with entities, attributes and relations," in *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI'16)*, 2016, pp. 2866–2872.

[17] J. Feng, M. Huang, M. Wang, M. Zhou, Y. Hao, and X. Zhu, "Knowledge graph embedding by flexible translation," in *Proceedings of the 15th International Conference on Principles of Knowledge Representation and Reasoning (KR'16)*, 2016, pp. 557–560.

[18] Y. Hao, Y. Zhang, S. He, K. Liu, and J. Zhao, *A Joint Embedding Method for Entity Alignment of Knowledge Bases*. Springer, 2016, pp. 3–14.

[19] P. Cai, W. Li, Y. Feng, Y. Wang, and Y. Jia, "Learning knowledge representation across knowledge graphs," in *AAAI 2017 Workshop on Knowledge-Based Techniques for Problem Solving and Reasoning (KnowProS'17)*, 2017.

[20] S. Araujo, D. T. Tran, A. P. de Vries, and D. Schwabe, "SERIMI: Class-based matching for instance matching across heterogeneous datasets," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 5, pp. 1397–1440, 2015.

[21] E. Jiménez-Ruiz and B. C. Grau, "Logmap: Logic-based and scalable ontology matching," in *Proceedings of the 10th International Conference on The Semantic Web - Volume Part I (ISWC'11)*, 2011, pp. 273–288.

[22] D. Ngo and Z. Bellahsene, "Overview of YAM++(not) yet another matcher for ontology alignment task," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 41, pp. 30–49, 2016.

[23] Y. Jia, Y. Wang, H. Lin, X. Jin, and X. Cheng, "Locally adaptive translation for knowledge graph embedding," in *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI'16)*, 2016, pp. 992–998.

[24] H. Lin, Y. Wang, Y. Jia, J. Xiong, P. Zhang, and X. Cheng, *An Ensemble Matchers Based Rank Aggregation Method for Taxonomy Matching*. Springer, 2015, pp. 190–202.

[25] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the 26th International Conference on Neural Information Processing Systems (NIPS'13)*, 2013, pp. 3111–3119.

[26] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph and text jointly embedding." in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP'14)*, vol. 14, 2014, pp. 1591–1601.