Unsupervised entity alignment using attribute triples and relation triples

Unsupervised Entity Alignment using Attribute Triples and Relation Triples

Fuzhen He[1,2], Zhixu Li[1,3⋆], Yang Qiang[4], An Liu[1], Guanfeng Liu[5], Pengpeng Zhao[1], Lei Zhao[1], Min Zhang[1], and Zhigang Chen[3]

[1]Institute of Artificial Intelligence, School of Computer Science and Technology, Soochow University, China
[2]Neusoft Corporation [3]IFLYTEK Research, Suzhou, China
[4]King Abdullah University of Science and Technology, Jeddah, SA
[5]Department of Computing, Macquarie University, Sydney, Australia
fzhe@stu.suda.edu.cn
{zhixuli, anliu, ppzhao, zhaol, minzhang}@suda.edu.cn
qiangyanghm@hotmail.com, guanfeng.liu@mq.edu.au, zgchen@iflytek.com

Abstract. Entity alignment aims to find entities referring to the same real-world object across different knowledge graphs (KGs). Most existing works utilize the relations between entities contained in the relation triples with embedding-based approaches, but require a large number of training data. Some recent attempt works on using types of their attributes in attribute triples for measuring the similarity between entities across KGs. However, due to diverse expressions of attribute names and non-standard attribute values across different KGs, the information contained in attribute triples can not be fully used. To tackle the drawbacks of the existing efforts, we novelly propose an unsupervised entity alignment approach using both attribute triples and relation triples of KGs. Initially, we propose an interactive model to use attribute triples by performing entity alignment and attribute alignment alternately, which will generate a lot of high-quality aligned entity pairs. We then use these aligned entity pairs to train a relation embedding model such that we could use relation triples to further align the remaining entities. Lastly, we utilize a bivariate regression model to learn the respective weights of similarities measuring from the two aspects for a result combination. Our empirical study performed on several real-world datasets shows that our proposed method achieves significant improvements on entity alignment compared with state-of-the-art methods.

Keywords: Unsupervised Entity Alignment · Interactive Model · Bivariate Regression Model · Relation Triples · Attribute Triples

1 Introduction

Entity alignment is a core task in knowledge graphs (KGs) integration, which aims to find entities referring to the same real-world object across different KGs.

⋆ The corresponding author

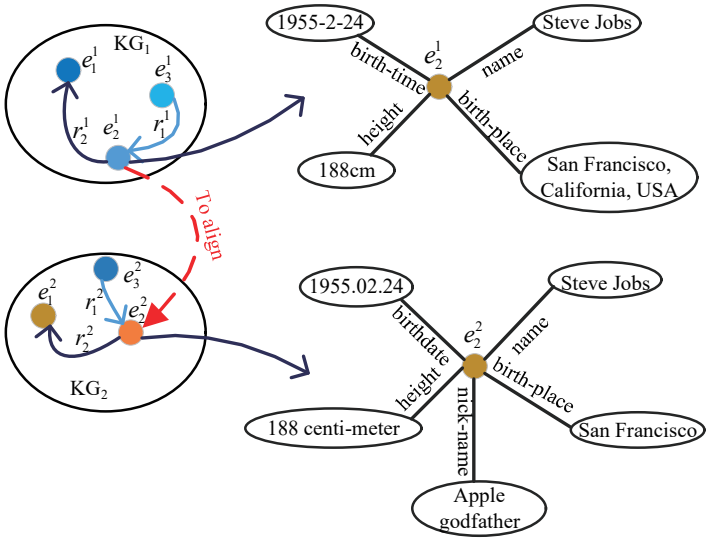


Fig. 1. An example for entity alignment between two KGs

It is pretty challenging due to the diverse expressions and structures of knowledge in different KGs. An example of entity alignment scenario is depicted in Fig. 1, where both e_2^1 from the KG_1 and e_2^2 from the KG_2 refer to *Steven Jobs*.

Most existing approaches for entity alignment tend to utilize the relations between entities on different KGs. Typically, they encode these entities and their relations with the other entities on KGs into a semantic space, such that the similarity between entities could be measured. Various embedding models, TransE [2] and its variants [12,13,19], have been proposed to perform the encoding. However, building the relation embedding model requires a large number of aligned entity pairs for training, which may not always be easily available. In addition, entity relations do not always have high quality (such as incompleteness or inaccuracy), which may harm the accuracy of the entity alignment results. Out of these reasons, some researchers turn to explore how to leverage crowdsourcing to achieve large-scale annotated data for training [21].

Some recent attempt utilizes the attributes of entities in entity alignment. An alignment model called JAPE (Joint Attribute-Preserving Embedding) [17] is proposed, which jointly embeds the structures of two KGs into a unified vector space and further refines it by leveraging the attribute correlations between KGs. To avoid from tackling the problem of diverse expressions of attribute values, JAPE does not fully use the information of attribute values, but simplifies the attribute values into data types, e.g., (Steven Jobs, birthdate, 1955-02-24) is simplified into (Steven Jobs, birthdate, Datetime). So far, no existing work considers to fully use attribute names and values for entity alignment given that both the expressions of attributes and attribute values are diverse across KGs. As can be observed in Fig. 1, an attribute "birth time" could be denoted by "birth-time" in KG_1 but "birthdate" in KG_2. There also exist a lot of non-standard

attribute values in KGs, e.g., a date time could be represented in several different formats such as "1955-02-24", "02/24/1955", and "24th Feb. 1955", etc.

To tackle the drawbacks of the existing approach, we propose an unsupervised entity alignment approach using both attribute triples and relation triples of KGs. Initially, we propose to fully use attribute triples for entity alignment, which will generate a lot of high-quality aligned entity pairs. We then use these aligned entity pairs to train a relation embedding model such that we could use relation triples to further align the remaining entities.

The challenge lies on how to fully use attribute triples for entity alignment. So as to tackle the two challenges mentioned above in using attribute triples for entity alignment, we novelly propose an interactive approach for entity alignment by performing entity alignment and attribute alignment alternately. It is worth noting that instead of performing entity alignment based on attribute names and values in one round, which deprives us further chances to update the alignment results, we make full use of the interaction between them to benefit each other in an iterative way. That is, we first perform entity alignment using values of common attributes and then do attribute alignment based on the matched entities, which could mutually promote the two alignment results iteratively.

Lastly, we utilize a bivariate regression model to learn respective weights of the similarity results, measured from the proposed iteration model using attribute triples and the relation embedding model using relation triples. In this way, we could fit the weights between the leveraging of relations and attributes. That is, on one hand, it is definitely beneficial for the alignment of entities with few relations in KGs to use attributes. On the other hand, taking advantage of relations could greatly reduce the harm brought by the inconsistent expressions of attributes names and values across different KGs.

We summarize our contributions as follows:

- We propose an unsupervised entity alignment approach using both attribute triples and relation triples of KGs, where the entity alignment results based on attribute triples could provide the training data for learning the relation embedding model based on relation triples.
- To deal with the challenges of diverse expressions of attributes names, we novelly propose an interactive approach for entity alignment by performing entity alignment and attribute alignment alternately.
- We utilize a bivariate regression model to fit the weights between the leveraging of relations and attributes in merging the alignment results of the two models.

Our empirical study conducted on several real-world data sets demonstrates that our entity alignment approach reaches a higher precision than the state-of-the-art relationship-embedding and attribute-embedding approaches.

Roadmap. The rest of the paper is organized as follows: We define the entity alignment problem in Sec. 2, and then present our approach in Sec. 3. After reporting our empirical study in Sec. 4, we cover the related work in Sec. 5. We finally conclude in Sec. 6.

2 Problem Definition

A typical *KG* consists of a number of *facts*, usually in the form of *triples* denoted by *(subject, predicate, object)*, where the *subject* is an *entity*, and the *object* can be either another *entity* or an attribute *value* of the subject entity. We call a triple as a *Relation Triple* if the *object* of the triple is also an entity and the *predicate* denotes the relation between the two entities. We call a triple as an *Attribute Triple* if the *predicate* denotes an attribute of the entity and the *object* of the triple is the corresponding attribute value of the entity.

Given two KGs, the task of entity alignment aims at identifying all pairs of entities referring to the same real-world objects between the two KGs. More formally, we define the entity alignment problem as follows.

Definition 1. (Entity Alignment). *Given two knowledge graphs denoted by* $KG_1 = \{E_1, RT_1, AT_1\}$ *and* $KG_2 = \{E_2, RT_2, AT_2\}$*, where* E_i*,* RT_i *and* AT_i *are the set of entities, relation triples and attribute triples of* KG_i *respectively* $(i = \{1, 2\})$*, the task of* ***Entity Alignment*** *aims at finding every entity pair* $\{(e_m^1, e_n^2)|m \in [1, |E_1|], n \in [1, |E_2|], e_m^1 \in E_1, e_n^2 \in E_2, e_m^1 \stackrel{\circ}{=} e_n^2\}$*, where* $e_m^1 \stackrel{\circ}{=} e_n^2$ *means* e_m^1 *and* e_n^2 *refer to the same real-world object.*

3 Our Approach

The architecture of our approach is given in Fig. 2: The inputs are two KGs, KG_1 and KG_2, and the outputs are aligned entity pairs. Our approach uses both relation triples and attribute triples for entity alignment, and then combines the two alignment results by using a bivariate regression model to learn the respective weights. In our approach, we propose to fully use attribute triples for entity alignment, which will generate a lot of high-quality aligned entity pairs. We then use these aligned entity pairs to train a relation embedding model(called structure embedding) such that we could use relation triples to further align the remaining entities. For using attribute triples for entity alignment, we novelly propose an interactive approach for entity alignment by performing entity alignment and attribute alignment alternately.

In the following of this section, we first introduce the interactive model using attribute triples for entity alignment in Sec. 3.1, and then employ the results of the interactive model on the structure embedding for the training set in Sec. 3.2. We finally present how we use the bivariate regression model for alignment results combination in Sec. 3.3.

3.1 An Interactive Model using Attribute Triples

A basic way to use attribute triples for entity alignment is to identify the percentage of common attributes and attribute values between the attribute triples of two entities, according to which we could measure a similarity between entities. However, the number of the common attributes with the same names across

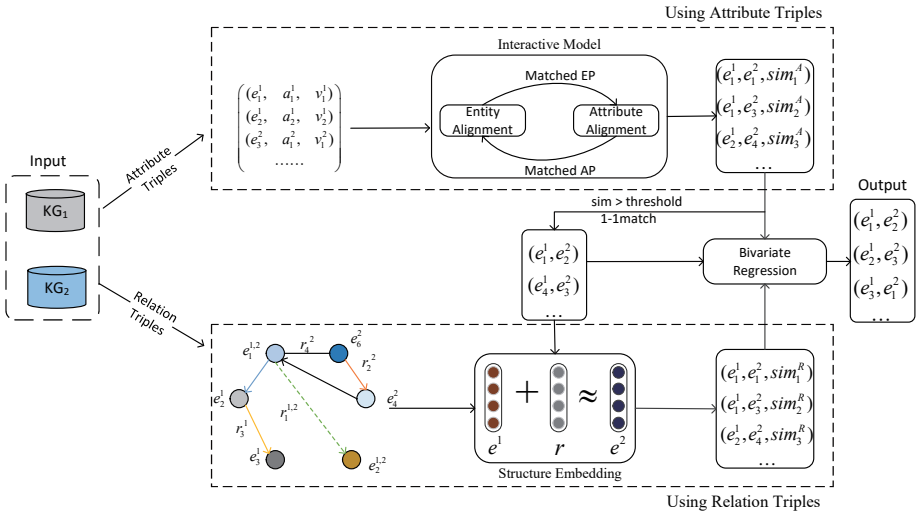


Fig. 2. Architecture of our approach

different KGs might be small. The diverse expressions of attribute values also aggravate the case. Thus it is very intractable for entity alignment with the basic way.

In this approach, we would like to perform attribute alignment together with entity alignment, based on the observation that the two tasks could be mutually reinforced by each other. That is, the matched entities are helpful to find more matched attributes and vice versa. Therefore, we propose an interactive model to make the two processes perform alternately.

Specifically, in each iteration, we first do entity alignment based on attribute values and build the matching set of entity pairs $\mathcal{O}_E$ increasingly and then employ the results to do attribute alignment and establish the matching set of attribute pairs $\mathcal{O}_A$ cumulatively. We iteratively repeat the above process until there are no more new common attributes or matched entities generated. The algorithm of interactive model is introduced in Algorithm 1. Next, we introduce the details of the interactive model as follows.

1) Entity Alignment based on Aligned Attributes. As mentioned before, we do entity alignment based on the common attributes between KGs. However, we find that there exists a huge gap in the same entities' attributes when analyzing data extracted from Baidu Encyclopedia and Wikipedia. This results from the following reasons: low coverage of attributes, various values of common attributes, and diversity of attribute names.

First, when using the values of attributes to do entity alignment, we find that the coverage of attributes is very low such that we cannot give the attributes different weights from a holistic view according to their importance, just like the way we do in relational databases. Thus, we have to repute that all common attributes of two entities share the equal weights. Based on this idea, we define

Algorithm 1: An Interaction Algorithm

Input : $KG_i = \{E_i, RT_i, AT_i\}, i = 1, 2$, where $E_i = \{e_1^i, e_2^i, ..., e_{|E^i|}^i\}$, RT_i, AT_i denote the Relation Triples and Attribute Triples. T_e, T_a denote the threshold of the accepted Entity Pairs and Attribute Pairs.

Output: Matching Entity Pairs

1. Set $\mathcal{O}_E = \mathcal{O}_A = \mathcal{O}_A^{iter} = \mathcal{O}_E^{iter} = \oslash$, where $\mathcal{O}_E, \mathcal{O}_A$ stand for all accepted matching Entity Pairs and Attribute Pairs, $\mathcal{O}_E^{iter}, \mathcal{O}_A^{iter}$ are the results of iter-th iteration. And let $iter = 0$;
2. **while** $\mathcal{O}_E^{iter} \neq \oslash$ *or* $\mathcal{O}_A^{iter} \neq \oslash$ *or* $iter = 0$ **do**
   3. $\mathcal{O}_E^{iter} = \mathcal{O}_A^{iter} = \oslash$;
   4. $\forall ep = (e_i^1, e_j^2), e_i^1 \in E_1, e_j^2 \in E_2$;
   5. **if** $Sim(e_i^1, e_j^2) \geq T_e$ **then**
      6. Add (e_i^1, e_j^2) to $\mathcal{O}_E^{iter}$,$\mathcal{O}_E$;
   end
   7. Extract two attribute sets $Attr_1, Attr_2$ from KG_1 and KG_2, among which are all belonging to the entities in $\mathcal{O}_E$;
   8. $\forall ap = (attr_i^1, attr_j^2), attr_i^1 \in Attr_1, attr_j^2 \in Attr_2$;
   9. **if** $Sim_A(attr_i^1, attr_j^2) \geq T_a$ **then**
      10. Add $(attr_i^1, attr_j^2)$ to $\mathcal{O}_A^{iter}$,$\mathcal{O}_A$;
      11. Replace $attr_i^1$ in KG_1 to $attr_j^2$;
   end
   12. $iter++$;
end
return $\mathcal{O}_E, \mathcal{O}_A$;

the following function to calculate the similarity between two entities:

$$Sim_A(e_1, e_2) = \frac{1}{n} \sum_{k=1}^{n} Sim_V(v_k^1, v_k^2) \tag{1}$$

where $Sim_A(e_1, e_2)$ denotes the similarity between entities e_1 and e_2, v_k^1 represents the e_1's value of the $k-th$ attribute owned by both e_1 and e_2, and n is the size of all same attributes. Sim_V denotes the similarity between attribute values, v_k^1 and v_k^2, which is defined as the following equation:

$$Sim_V(v_k^1, v_k^2) = \frac{lcsSim(v_k^1, v_k^2)}{levenshteinSim(v_k^1, v_k^2) + lcsSim(v_k^1, v_k^2)} \tag{2}$$

where $levenshteinSim(\cdot, \cdot)$ is Levenshtein distance [8], a string metric for measuring the difference between two sequences. $lcsSim(\cdot, \cdot)$ measures the similarity through two strings' Longest Common Substring [9]. As a result, we can make full use of entities' common attributes to represent the entities' similarity.

Second, we also find that although entities have the common attributes, the values often vary greatly, especially for numeric attributes, e.g., the form of a person's birth time may show as "1955-02-24", "24/02/1955", or "24th Feb. 1955 ", etc. If calculating their similarity through $Sim_V(\cdot, \cdot)$, the results will be rather inaccurate. Moreover, due to different statistical time, the values of the number

vary a lot, e.g., the values of attribute "population" for the same city in different datasources are different in different years. So as to tackle these problems, we normalize values into the most popular ones, e.g., "yyyy-mm-dd" for date. Then we extract all numbers in string through regular expression and define the following function to calculate their values' similarity through extracting number from the normalized values:

$$Sim(num_1, num_2) = \frac{|num_1 - num_2|}{max(num_1, num_2) + 1} \tag{3}$$

And only if $Sim(num_1, num_2) \leq 0.01$, num_1 will be seen as equivalence to num_2. Furthermore, we normalize some other attributes like area, length, height, population.

Last but not the least, considering the case when two entities share no common attributes, we are unable to calculate their similarity even if they refer to the same real-world object. In addition, the diversity of attribute names across KGs makes the phenomenon quite serious, such as the attribute "birth time" displaying as "birthdate" in Baidu Encyclopedia but presenting as "birth-time" in Wikipedia. We cannot use this kind of attributes well for entity alignment. Therefore, we decide to utilize the aligned entities to align this kind of attributes. The details will be introduced in the following part.

2) Attribute Alignment based on Aligned Entities. As mentioned before, we can find more possibly aligned attribute pairs by leveraging the set of aligned entity pairs. Supposing that in a certain iteration *iter*, we have a set of aligned entity pairs, $\mathcal{O}^E_{iter} = \{ep_1, ep_2, ep_3, ..., ep_{|\mathcal{O}^E_{iter}|}\}$ where ep_i is the aligned entity pair (e^1_i, e^2_j). Next, for the candidate attribute pairs (a^1_m, a^2_n) which are possed by the subset of $\mathcal{O}^E_{iter}$ denoted as $EP_{(a^1_m, a^2_n)} = \{ep_1, ep_4, ep_{10}, ..., ep_k\}$. We leverage all values of attributes owned by entity pairs to represent the similarity of attribute pairs. For each $ep_l (1 \leq l \leq k)$ whose attribute values are v^i_m and v^i_n for attribute pair (a^1_m, a^2_n), the similarity can be calculated with the following equation:

$$Sim_A(a^1_m, a^2_n) = \frac{1}{Z} \sum_{i=1}^{Z} Sim_V(v^i_m, v^i_n) \tag{4}$$

where $Z = |EP_{\{a^1_m, a^2_n\}}|$ represents the size of $EP_{(a^1_m, a^2_n)}$ and $SimV(\cdot, \cdot)$ denotes the similarity between attribute values. Here we take a running example to illustrate our model as follows.

Example 1. As we can see in Fig. 3, we suppose the precondition is $a^1_i = a^2_i, i \in [1, 5]$ and we do not know the matching results of the remaining attributes. Under the precondition, each entity pair has at least one common attribute except (e^1_3, e^2_4). Through leveraging these attribute alignment seeds, we can get two aligned entity pairs (e^1_1, e^2_1) and (e^1_2, e^2_2), but they are not able to determine whether e^1_3 and e^2_4 are aligned. Then, based on the two aligned entity pairs, we do attribute alignment. Two more attribute matching pairs (a^1_6, a^2_6) and (a^1_8, a^2_8) are generated. In the next iteration, leveraging the attribute alignment results gen-

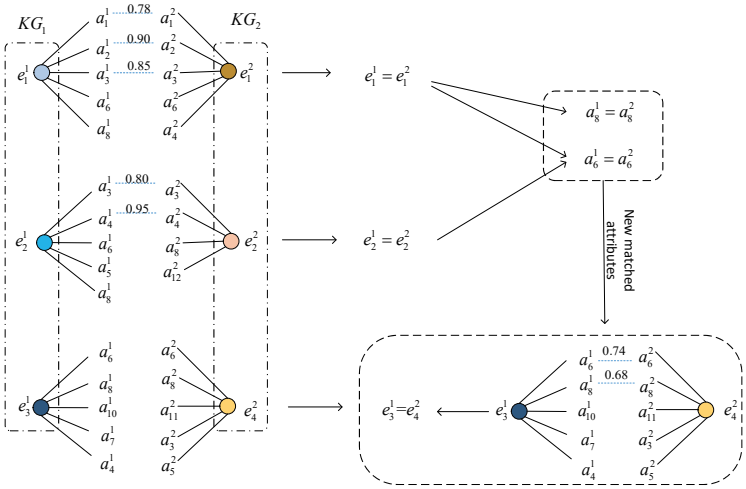


Fig. 3. An example of interactive alignment model

erated in the first iteration, we can determine the matching result of the entity pair (e_3^1, e_4^2)*, which is aligned.*

3.2 Structure Embedding

We select entity pairs with high confidence (larger than a predefined threshold) from the results of interactive model as the training set for structure embedding. Then we use relation triples and the training set to do structure embedding (SE), aiming to model the geometric structures of two KGs and learning approximate representations for entities and relations. Formally, given a relation triple $tr = (h, r, t)$ where h is the head entity and r is the tail entity, we expect $h + r = t$. To measure the plausibility of tr, SE model optimizes the margin-based ranking loss to make the scores of positive triples lower than negative ones [17]:

$$\mathcal{O}_{SE} = \sum_{tr \in Tr} \sum_{tr' \in Tr'} (f(tr) - \alpha(tr')) \tag{5}$$

where $f(tr) = ||h+r-t||_2^2$ is the score function, Tr and Tr' are the sets of all positive triples and the associated negative triples, and α is a ratio hyper-parameter that weights positive and negative triples and its range is [0, 1]. Through the above embedding process, we could learn approximate vector representations of the entities across KGs, e.g., in order to get the similarity between an entity pair (e_i, e_j), we can calculate its similarity with entities' embedding results $(\boldsymbol{e_i}, \boldsymbol{e_j})$ according to the following equation:

$$simR(e_i, e_j) = \cos(\boldsymbol{e_i}, \boldsymbol{e_j}) \tag{6}$$

3.3 A Bivariate Regression Model for Weights Allocation

As we said before, we represent the similarity between entities both from the aspects of relations and attributes. Specifically, we incorporate the similarities calculated by the relation triples and attribute triples in linear weighting. We utilize the following formulation to represent the final similarity of any entity pair $e_i \in KG_1$ and $e_j \in KG_2$:

$$Sim(e_i, e_j) = \lambda \cdot simR(e_i, e_j) + (1 - \lambda) \cdot simA(e_i, e_j) \tag{7}$$

where $simR(\cdot, \cdot)$ is the similarity measure to calculate the embedding similarity using relation triples, and $simA(\cdot, \cdot)$ represents the similarity of entities calculated through their attribute triples. λ is the parameter to balance the importance of left part and right part.

Instead of setting parameter λ artificially, we take the aligned entity pairs as training data to learn λ through a bivariate regression model. It comes from the fact that for different datasets, the respective importance of relations and attributes should be different, i.e., for a data set full of entities with high-quality relations, the weight of relations should have a higher confidence. On the contrary, if the number of entities' relations is quite small, attributes should be assigned with a higher weight. Specifically, we leverage those training entity pairs, also treating as the training set of *Unsupervised SE* model, to construct the input of our regression model. e.g., for a matched entity pair (e_i, e_j), we suppose it's final similarity to be 1.0. So we let the input form be $(simR(e_i, e_j), simA(e_i, e_j), 1.0)$. We want the final similarities $Sim(e_i, e_j)$ of these matching entity pairs close to 1.0 as much as possible. Besides, we use MSE(Mean Squared Error) [20] as our loss function and SGD [3] as the optimizer.

4 Experiments

We first introduce our datasets and the metrics we would use for evaluation, and then represent the existing state-of-the-art approaches that we would compare with. Finally, we evaluate our proposed approach on several metrics.

4.1 Datasets and Metrics

We collect our data from Baidu Encyclopedia [1], Wikipedia [2] and Hudong Encyclopedia [3]. We extract triples from the infobox, if a value in the infobox is a link or contains several links, we will save it as a relation triple or separate it into several relation triples. Otherwise, we take it as an attribute triple. Then, we do object linking for all triples to replace the values with the URIs of the entities that they actually refer to in the KG. Specially, for some relation triples,

[1] https://baike.baidu.com
[2] https://zh.wikipedia.org
[3] http://www.baike.com

whose objects are not in our KG, we treat them as attribute triples, aiming to construct a small but complete KG. Here, we have two datasets where one is about persons entities, places entities and others, while the other is about natural creatures, like animals and plants. As we can see in Table 1, the number of attribute triples are nearly 4 times of relation triples in the first dataset while the number of relation triples and attribute triples are almost equal in the second dataset.

Table 1. Statistics of the datasets

Datasets		**Entities**	**Relations**	**Attributes**	**Rel.triples**	**Attr.triples**
dataset1	Baidu	12,647	715	5,166	29,373	108,052
	Wiki	8,218	231	1,904	14,351	66,249
dataset2	Baidu	13,983	470	2,322	79,025	79,658
	Hudong	10,263	79	870	44,613	56,270

As for the evaluation metrics, we use $Hits@k$ and $Mean$ to assess the performance of the approaches. $Hits@k$ reflects the proportion of correctly aligned entities ranked in the top k. And $Mean$ calculates the mean of these ranks. Naturally, a higher $Hits@k$ and a lower $Mean$ indicate a better performance. We also evaluate the influence on the precision of the iterative time in the interactive model.

4.2 Approaches for Comparison

In this section, we briefly introduce five comparative methods including *SE* [2], *JAPE* [17], our proposed *Baseline Model*, the *Interactive Model* and *Interactive Model + Unsupervised SE*.

- The *SE* method aims to learn representations of all entities and relations in KG_1 and KG_2. In order to model the geometric structures of two KGs, *SE* serves the seed alignments as bridge to build an overlay relationship graph, essentially encoding the entities and relationships of various KGs into a semantic space to measure the similarity between entities.
- The *JAPE* method jointly embeds the structures of two KGs into a unified vector space and further refines it by leveraging attribute correlations in the KGs. It employs two models, namely structure embedding (SE) and attribute embedding (AE), to learn embedding of KGs where SE models relation structures of two KGs from the aspect of relation triples and AE attempts to encode the entities more accurately leveraging attribute triples.
- The *Baseline* only leverages the attribute values to do entity alignment just one time and attribute alignment is not involved.
- The *Interactive Model* (*IM* for short) also just considers the attribute values, but different from *Baseline*, it also lets entity alignment and attribute alignment benefit each other iteratively.

– The *Interactive Model + Unsupervised SE* (*IMUSE* for short) method takes some high-quality entity pairs generated by the *Interactive Model* as the training data of *Unsupervised SE* model, and then incorporates results of two models by linear weighting.

4.3 Experimental Results

1) Top-K and Mean Results. For *SE* and *JAPE*, we use 60% entity pairs of the groundtruth as alignment seeds while the rest is the testing data. Note that the other three methods do not need these seeds for training. The predefined thresholds are 0.89 and 0.92 respectively for dataset1 and dataset2. As can be seen from Table. 2, when varying *hits*@*k*, *IMUSE* always largely outperforms the other methods on the two datasets in that it incorporates relation embedding method with attribute-value based method which can find more aligned entities. And *JAPE* performs better than *SE* since it encodes entities through jointly embedding the relation and attribute triples. And we can see that the iterative process enables *IM* perform better than *Baseline*, because *IM* can benefit from the interaction to align more entities. We can also see that in Table 2, *IMUSE*

Table 2. Comparison with Hits@k and Mean

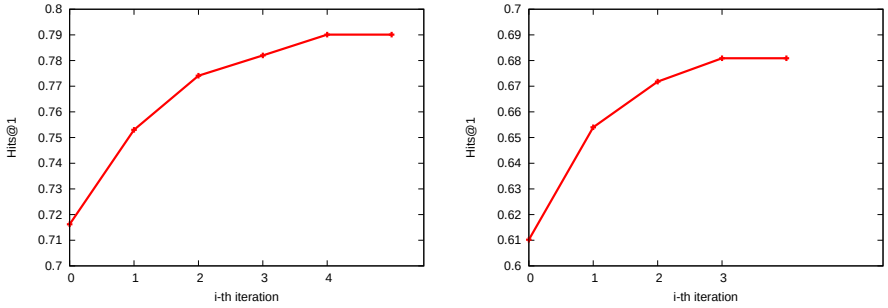
Approaches	Dataset1				Dataset2			
	Hits@1	Hits@5	Hits@10	Mean	Hits@1	Hits@5	Hits@10	Mean
SE	0.5825	0.6539	0.7464	29	0.6331	0.6570	0.6982	58
JAPE	0.6317	0.7182	0.8157	17	0.7029	0.7362	0.7451	30
Baseline	0.7162	0.9448	0.9682	8	0.6102	0.8943	0.9270	14
IM	0.7901	0.9590	0.9747	5	0.6809	0.9274	0.9416	10
IMUSE	**0.8232**	**0.9593**	**0.9755**	**3**	**0.7336**	**0.9328**	**0.9560**	**7**

has the smallest Mean values among *IM* and *Baseline* in two datasets while *SE* and *JAPE* have the highest Mean values.

It is noticeable that the results of these models on the two datasets perform a little difference on *Hits*@1, which results from the following reasons. (1) Entities in the first dataset have an average of 2 relations and 8 attributes while entities in the second dataset have 5 relations and 5 attributes averagely. Therefore, *SE*, only using entities' relation information, performed worse than *Baseline* and *IM* on the dataset1 but better on the dataset2. (2) As is shown in Table 1, the kinds of relations and attributes are much smaller in the dataset2 than in the dataset1. So, these methods performed a little worse on the dataset2. (3) Entities in the dataset2 are mainly about animals and plants, whose properties(relations and attributes) are almost "Kingdom", "Phylum", "Classes", "Family", etc, and the values under these properties are very similar. So, the final result of dataset2 is not so good as that in the dataset1.

2) The Effectiveness of Iterations. We also evaluate the influence of iterations on *Hits@1* for our proposed method *IMUSE*. As shown in Fig 4, with the increase of iterative time, *Hits@1* also went up with it since new aligned

attributes contribute to aligning more entities and vice versa. In addition, when the iterative time increases to 4 and 3 for the dataset1 and the dataset2 respectively, *Hits@1* of them keep stable in that no more entities or attributes can be aligned.



(a) Dataset1 (b) Dataset2

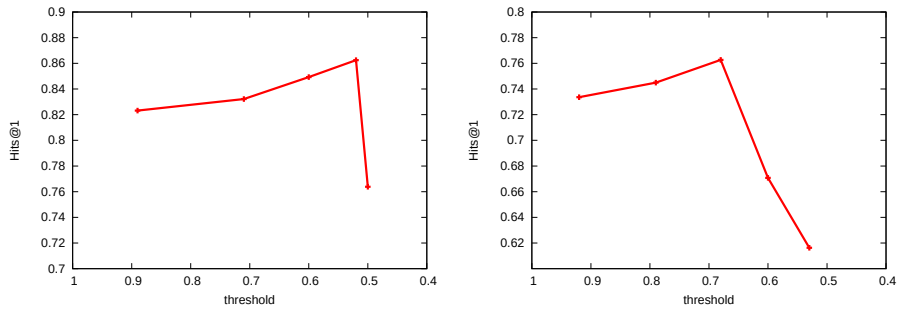
Fig. 4. The Effectiveness of Iteration

Table 3. The effectiveness of Dynamic Combination

	SE	IM	Static Combination	Dynamic Combination
Dataset1	0.5825	0.7901	0.8127	**0.8232**
Dataset2	0.6331	0.6809	0.7110	**0.7336**

3) Dynamic Combination (Bivariate Regression) v.s. Static Combination. Instead of combining the results generated by *Interactive Model* and *Unsupervised SE* in a static way, we choose to learn respective weights of similarities measuring from the aspects of relations and attributes through bivariate regression, called *Dynamic Combination*. On the contrary, we manually give the two results the same weights to combine them together to represent the process of *Static Combination*. As is shown in Table 3, *Dynamic Combination* truly performs better than *Static Combination* because learning weights contain the information of different importance of the entities' relations and attributes when doing entity alignment.

4) The Effectiveness of Different Predefined Thresholds. As mentioned before, the predefined threshold enables us to select entity pairs with high confidence as the training set for structure embedding. Obviously, the higher the threshold is, the higher accuracy that the training set is, but the less training entity pairs it gets. Intuitively, there may be a balance between the accuracy and quantity. In order to find the optimal result, according to the size of training set, we set five different thresholds, corresponding to 30%, 40%, 60%, 80% and 90% of the size of SE's training set, to illustrate the idea on two datasets.



(a) Dataset1 (b) Dataset2

Fig. 5. The Effectiveness of Different Thresholds

As we can see in Fig. 5, on dataset1 when the threshold decreases, the value of hit@1 goes up. When the threshold is 0.52, the accuracy is the highest, 86.25%. After that, it begins to drop. Similarly, on the dataset2 the accuracy reaches the best result, 76.27%, when the threshold is 0.68. Therefore, we can conclude that the predefined threshold truly plays an important role to the final results.

5 Related Work

Entity alignment is a sub-problem of KG integration. In this section, we first give some introduction to a similar problem of KG integration called database integration, and then cover some mainstream methods on entity alignment.

5.1 Database Integration v.s. KG Integration

Data integration in relational databases has gained lots of attentions [11], which consists of two main tasks, i.e., Record Matching [5] and Schema Matching [16]. Record Matching aims at identifying records in the same or different databases that refer to the same real-world objects.

There are several typical approaches commonly used to detect approximately duplicate records such as probabilistic methods [14], supervised or unsupervised machine learning techniques [15, 18], variations based on active learning [4], etc. Schema Matching refers to matching combinations of elements that appear together in a structure. In current implementations, schema matching is typically performed manually, which has significant limitations. Of course, some automatic matching algorithms have been proposed based on name similarity [1], description similarity [10], etc. Besides, some other methods [6] incorporate crowdsourcing to improve the accuracy of automatic algorithms.

Compared to the relational database integration, KG integration is more challenging due to the complex structures of KGs and more diverse expressions of properties and values. In relational databases, entities belonging to the same field always appear in the same table. Even if some difficulties, like transcription

errors and incomplete information, actually exist, these entities' properties are all same to each other. However, due to diverse expressions of properties and values in KGs, only a few entities have identical properties. As a result, entity alignment in KGs are faced with more challenges.

5.2 Entity Alignment

Entity alignment is pretty challenging due to the diverse expressions and structures of knowledge in different KGs. So far, plenty of work has been done on this problem, most of which are based on embedding methods. Some of them only leverage relation triples, commonly called KG Embedding, while the others jointly embed both relation and attribute triples, like JAPE [17].

KG Embedding. Recently, a lot of researchers have been trying every effort to learning and improving KG embedding. For example, TransE [2], the basic of all embedding based methods, treats a relation triple as *(h, r, t)* where h is head entity, r is relation, and t is tail entity. TransE tries to embed entities and relations of multi-relational data to low-dimensional vector spaces. So as to decrease the costs of training and reduce the number of parameters, Bordes et al. interpret relations as translating operations from head to tail entities, which can be expressed as $\mathbf{h} + \mathbf{r} = \mathbf{t}$. Since then, a lot of embedding models have been proposed to improve TransE, such as TransR [13] and PTransE [12]. Different from TransE, they project both entities and relations into a continuous vector space. Specifically, TransR model maps entities and relations into separated entity space and relation space, and performs translation in relation space. Considering that TransE only leverages individual triples and ignores multi-step relation paths, PTransE [12] encodes multi-step paths to address this issue. However, most of these methods just focus on how to encode relation triples in better ways, neglecting those attribute triples. Especially for the entities lacking in relations, it does not work well if only leveraging relations.

JAPE. In order to take advantage of attribute triples, Sun et al. propose an alignment model called JAPE (Joint Attribute-Preserving Embedding) which focuses on the cross-lingual entity alignment [7]. It jointly embeds the structures of two KGs and further refines it by leveraging attribute correlations in the KGs. Specifically, it employs two models, namely structure embedding (SE) and attribute embedding (AE), to learn embedding of KGs where SE models relation structures of two KGs from the aspect of relation triples and AE attempts to cluster those attributes often used together to describe an entity. The average of attributes embedding results will be used to encode entities. As a result, those entities which possess similar attributes will be close to each other. Finally, it combines SE and AE to jointly embed all the entities in two KGs into a unified vector space. However, JAPE does not use the information of attribute values well. Alternatively, so as to tackle the problem of diverse expression of values, they reduce the attribute values to data type, e.g., (Barack Obama, birthdate, 1961-08-04) is replaced by (Barack Obama, birthdate, Datetime). Obviously, it is just a coarse-grain way to deal with attribute values where the importance of attribute values is ignored. In order to overcome the heterogeneity between

different KGs, we propose an interactive approach to integrate entity alignment and attribute alignment together.

6 Conclusions and Future Work

We work on leveraging both relation and attribute triples to do entity alignment between two KGs through our proposed model. In order to make full use of attribute triples, we propose an interactive method to do entity alignment and attribute alignment to calculate the similarities. And then we utilize the results of the interaction as the training set to embed relation triples for computing the similarities in an unsupervised manner. Last, we incorporate the two kinds of results in linear weights to represent the final similarities of entity pairs through a bivariate regression model. Our experiments on real-world datasets demonstrate that our approach performs much better than the state-of-the-art methods.

In the future work, we look forward to decreasing the influence of non-standard attribute values by learning patterns to build standard data forms. In addition, we would like to turn to crowdsourcing to help decide whether attribute pairs with great difference in expressions should be aligned.

Acknowledgments

This research is partially supported by National Natural Science Foundation of China (Grant No. 61632016, 61572336, 61572335, 61772356), the Natural Science Research Project of Jiangsu Higher Education Institution (No. 17KJA520003, 18KJA520010), and the Open Program of Neusoft Corporation (No. SKLSAOP1801).

References

1. Bell, G.B., Sethi, A.: Matching records in a national medical patient index. Communications of the ACM 44(9), 83–88 (2001)
2. 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)
3. Bottou, L.: Large-scale machine learning with stochastic gradient descent. In: Proceedings of COMPSTAT'2010, pp. 177–186. Springer (2010)
4. Cohn, D., Atlas, L., Ladner, R.: Improving generalization with active learning. Machine learning 15(2), 201–221 (1994)
5. Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate record detection: A survey. IEEE Transactions on knowledge and data engineering 19(1), 1–16 (2007)
6. Fan, J., Lu, M., Ooi, B.C., Tan, W.C., Zhang, M.: A hybrid machine-crowdsourcing system for matching web tables. In: Data Engineering (ICDE), 2014 IEEE 30th International Conference on. pp. 976–987. IEEE (2014)
7. Fu, B., Brennan, R., OSullivan, D.: Cross-lingual ontology mapping–an investigation of the impact of machine translation. In: Asian Semantic Web Conference. pp. 1–15. Springer (2009)

8. Heeringa, W.J.: Measuring dialect pronunciation differences using Levenshtein distance. Ph.D. thesis, Citeseer (2004)
9. Hirschberg, D.S.: A linear space algorithm for computing maximal common subsequences. Communications of the ACM 18(6), 341–343 (1975)
10. Larson, J.A., Navathe, S.B., Elmasri, R.: A theory of attributed equivalence in databases with application to schema integration. IEEE Transactions on software engineering 15(4), 449–463 (1989)
11. Lenzerini, M.: Data integration: A theoretical perspective. In: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. pp. 233–246. ACM (2002)
12. Lin, Y., Liu, Z., Luan, H., Sun, M., Rao, S., Liu, S.: Modeling relation paths for representation learning of knowledge bases. arXiv preprint arXiv:1506.00379 (2015)
13. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: AAAI. vol. 15, pp. 2181–2187 (2015)
14. Palopoli, L., Saccá, D., Terracina, G., Ursino, D.: A unified graph-based framework for deriving nominal interscheme properties, type conflicts and object cluster similarities. In: Cooperative Information Systems, 1999. CoopIS'99. Proceedings. 1999 IFCIS International Conference on. pp. 34–45. IEEE (1999)
15. Perkowitz, M., Doorenbos, R.B., Etzioni, O., Weld, D.S.: Learning to understand information on the internet: An example-based approach. Journal of Intelligent Information Systems 8(2), 133–153 (1997)
16. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. the VLDB Journal 10(4), 334–350 (2001)
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. Verykios, V.S., Elmagarmid, A.K., Houstis, E.N.: Automating the approximate record-matching process. Information sciences 126(1-4), 83–98 (2000)
19. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: AAAI. vol. 14, pp. 1112–1119 (2014)
20. Wang, Z., Bovik, A.C.: Mean squared error: Love it or leave it? a new look at signal fidelity measures. IEEE signal processing magazine 26(1), 98–117 (2009)
21. Yang, J., Fan, J., Wei, Z., Li, G., Liu, T., Du, X.: Cost-effective data annotation using game-based crowdsourcing. Proceedings of the VLDB Endowment 12(1), 57–70 (2018)