Multi-view Knowledge Graph Embedding for Entity Alignment

Qingheng Zhang\textsuperscript{1,*}, Zequn Sun\textsuperscript{1,*}, Wei Hu\textsuperscript{1,†}, Muhao Chen\textsuperscript{2}, Lingbing Guo\textsuperscript{1}, Yuzhong Qu\textsuperscript{1}

\textsuperscript{1} State Key Laboratory for Novel Software Technology, Nanjing University, China
\textsuperscript{2} Department of Computer Science, University of California, Los Angeles, USA

\{qhzhang, zqsun, lbguo\}@nju.edu.cn, \{whu, yzqu\}@nju.edu.cn, muhaochen@ucla.edu

Abstract

We study the problem of embedding-based entity alignment between knowledge graphs (KGs). Previous works mainly focus on the relational structure of entities. Some further incorporate another type of features, such as attributes, for refinement. However, a vast of entity features are still unexplored or not equally treated together, which impairs the accuracy and robustness of embedding-based entity alignment. In this paper, we propose a novel framework that unifies multiple views of entities to learn embeddings for entity alignment. Specifically, we embed entities on the views of entity names, relations and attributes, with several combination strategies. Furthermore, we design some cross-KG inference methods to enhance the alignment between two KGs. Our experiments on real-world datasets show that the proposed framework significantly outperforms the state-of-the-art embedding-based entity alignment methods. The selected views, cross-KG inference and combination strategies all contribute to the performance improvement.

1 Introduction

Entity alignment, a.k.a. entity matching or resolution, aims to find entities in different knowledge graphs (KGs) referring to the same real-world identity. It plays a fundamental role in KG construction and fusion, and also supports many downstream applications, e.g., semantic search, question answering and recommender systems. Conventional methods for entity alignment identify similar entities based on the symbolic features, such as names, textual descriptions and attribute values. However, the computation of feature similarity often suffers from the semantic heterogeneity between different KGs [Isele and Bizer, 2013]. Recently, increasing attention has been paid to leveraging the KG embedding techniques for addressing this problem, where the key idea is to learn vector representations (called embeddings) of KGs and find entity alignment according to the similarity of the embeddings. The vector representations can benefit the task of learning similarities [Wu et al., 2018]. Although existing embedding-based entity alignment methods have achieved promising results, they are still challenged by the following two limitations.

First, entities in KGs have various features, but the current embedding-based entity alignment methods exploit just one or two types of them. For example, MTransE [Chen et al., 2017], IPTransE [Zhu et al., 2017] and BootEA [Sun et al., 2018] only embed the relational structure of KGs for entity alignment, in addition to which JAPE [Sun et al., 2017], KD-CoE [Chen et al., 2018] and AttrE [Trsedya et al., 2019] complement attributes, textual descriptions or literals, respectively, to refine the embeddings. Actually, different types of features characterize different aspects of entity identities. Making use of them together would improve the accuracy and robustness.

Second, the existing embedding-based entity alignment methods rely on abundant seed entity alignment as labeled training data. However, in practice, such seed entity alignment is not always accessible and very costly to obtain [Isele and Bizer, 2013]. Furthermore, it is widely-acknowledged that entity alignment can benefit from relation and attribute alignment [Suchanek et al., 2012]. Still, the existing methods [Zhu et al., 2017; Trsedya et al., 2019] assume that seed relation and attribute alignment can be easily found beforehand. In fact, learning embeddings from various features would enable the automatic population of more alignment information and relieve the reliance on the seed alignment.

To cope with the above limitations, we propose MultiKE, a new entity alignment framework based on multi-view KG embedding. The underlying idea is to divide the various features of KGs into multiple subsets (called views), which are complementary to each other (see Figure 1 for example). So entity embeddings can be learned from each particular view.
and jointly optimized to improve the alignment performance. In summary, our main contributions are listed as follows:

- Based on the data model of KGs, we define three representative views based on the name, relation and attribute features. For each view, we employ an appropriate model to learn embeddings from it. (Section 3)
- For entity alignment, we design two cross-KG identity inference methods at the entity level as well as the relation and attribute level, respectively, to preserve and enhance the alignment between different KGs. (Section 4)
- We present three different strategies to combine multiple view-specific entity embeddings. Finally, we find entity alignment by the combined embeddings. (Section 5)
- Our experiments on two real-world datasets show that MultiKE largely outperforms existing embedding-based entity alignment methods. The selected views, cross-KG inference and combination strategies all contribute to the improvement. MultiKE also achieves promising results on unsupervised entity alignment and is comparable to conventional entity alignment methods. (Section 6)

2 Related Work

KG embedding. Learning KG embeddings has drawn much attention in recent years. Current KG embedding models can be divided into three kinds. Translational models, such as TransE [Bordes et al., 2013], TransH [Wang et al., 2014] TransR [Lin et al., 2015] and TransD [Ji et al., 2015], interpret a relation as a translation vector from a head entity to a tail. Semantic matching models use similarity-based functions to infer relation facts, e.g., the Hadamard product in DistMult [Yang et al., 2015] and ComplEx [Trouillon et al., 2016], and the circular correlation in HoIE [Nickel et al., 2016]. Neural models exploit deep learning techniques for KG embedding, e.g., multilayer perceptrons in ProjE [Shi and Weninger, 2017], convolutional neural networks in ConvE [Dettmers et al., 2018], and graph convolutional networks in R-GCN [Schlichtkrull et al., 2018]. All these models focus on relation facts and are mostly evaluated by the task of link prediction in a single KG.

Embedding-based entity alignment. The current methods represent different KGs as embeddings and find entity alignment by measuring the similarity between these embeddings. MTransE [Chen et al., 2017] learns a mapping between two separate KG embedding spaces. IPTransE [Zhu et al., 2017] and BootEA [Sun et al., 2018] are two self-training methods, which embed two KGs in a unified space and iteratively label new entity alignment as supervision. GCN-Align [Wang et al., 2018b] employs graph convolutional networks to model entities based on their neighborhood information. These methods align entities mostly based on the relation features of KGs. Additionally, several other methods leverage an additional type of features to enhance the relation-based embeddings. JAPE [Sun et al., 2017] models the attribute correlations with SkipGram, KDCoE [Chen et al., 2018] co-trains the embeddings of textual descriptions, and AttrE [Trsedya et al., 2019] conducts character-level literal embeddings. However, all these features are not equally treated but used to refine the relation-based embeddings. More importantly, these methods are incapable of incorporating new features.

Multi-view representation learning. Learning representations from multi-view data can achieve strong generalization performance. Recently, multi-view representation learning has been widely used in the network embedding [Qu et al., 2017; Wang et al., 2018a] and natural language processing (NLP) [Clark et al., 2018]. A typical process of multi-view representation learning is constituted by three major steps: (i) identify multiple views that can sufficiently represent the data; (ii) carry out representation learning on each view; and (iii) combine the multiple view-specific representations. We refer interested readers to [Li et al., 2016] for more details.

3 Multi-view KG Embedding

3.1 Problem Statement

In this paper, we study multi-view KG embedding for entity alignment, which aims to learn comprehensive entity embeddings based on different views. Particularly, for entity alignment, we consider three views for an entity identity, namely the name view, the relation view and the attribute view. Names describe a basic and key characteristic of entities. In KGs, entity names are usually described, for example, by rdfs:label, or the local names of entity URIs\(^1\). The relations and attributes defined in KG schemata characterize entities with rich extrinsic and intrinsic information. Based on the view division above, we formalize a KG as a 7-tuple \(G = (E, R, A, V, N, X, Y)\), where \(E, R, A\) and \(V\) denote the sets of entities, relations, attributes and literals, respectively. \(N \subseteq E \times V\) denotes the name view of entities, \(X \subseteq E \times R \times E\) denotes the relation view, and \(Y \subseteq E \times A \times V\) denotes the attribute view.

Given a source KG \(G_a = (E_a, R_a, A_a, V_a, N_a, X_a, Y_a)\) and a target KG \(G_b = (E_b, R_b, A_b, V_b, N_b, X_b, Y_b)\), entity alignment aims to find a set of identical entities \(M = \{e_i, e_j \in E_a \times E_b \mid e_i \equiv e_j\}\), where “\(\equiv\)” denotes the equivalence relationship. We first learn \(d\)-dimensional entity embeddings and then find entity alignment based on the embeddings.

For notations, we use bold lowercase letters to represent embeddings and bold uppercase letters to matrices. The superscript with bracket of an entity embedding indicates from which view this embedding comes. The name, relation and attribute views are marked by \((1)\), \((2)\) and \((3)\), respectively.

3.2 Literal Embedding

Literals are constituted by sequences of tokens. As the basis for multi-view embedding, literal embedding represents the discrete and symbolic literals by \(d\)-dimensional embeddings. Without loss of generality, let \(l = (o_1, o_2, \ldots, o_n)\) denote a literal of \(n\) tokens. \(LP(\cdot)\) is defined as a lookup function that maps the input to an embedding. To resolve the different expressions of literals and capture their inherent semantics, we consider the following cases:

\[
LP(o_i) = \begin{cases} \text{word} \_ \text{emb}(o_i) & \text{if } o_i \text{ has a word embedding} \\ \text{char} \_ \text{emb}(o_i) & \text{otherwise} \end{cases}, \quad (1)
\]

\(^1\)A local name is the string after the last hash or slash of a URI.
where word_embed(·) returns the corresponding word embedding of the input token. In fact, word embeddings have become a fundamental resource for many NLP applications, due to their capability of capturing semantic relatedness of words. Here, we use pre-trained word embeddings [Mikolov et al., 2018] to collocate similar literals. char_embed(·) returns the average of the character embeddings that are pre-trained with the Skip-gram model [Mikolov et al., 2013] on the KG literal set \( \mathcal{V}_a \cup \mathcal{V}_b \). Let \( \phi(·) \) be the literal embedding of the input. We employ an autoencoder to encode a list of token embeddings into one literal embedding in an unsupervised manner:

\[
\phi(l) = \text{encode}([\text{LP}(o_1); \text{LP}(o_2); \ldots; \text{LP}(o_n)]),
\]

(2)

where \( \text{encode}(·) \) returns the compressed representation of the input embeddings and \( [·] \) denotes the concatenation operation. We limit the maximum number of tokens to 5. Long literals are truncated while short ones are appended with placeholders.

3.3 Name View Embedding

We embed the name view using the above literal embeddings. Given an entity \( h \), its name embedding is defined as follows:

\[
h^{(1)} = \phi(\text{name}(h)),
\]

(3)

where \( \text{name}(·) \) extracts the name of the input object. We refer to the KG embeddings under the name view as \( \Theta^{(1)} \).

3.4 Relation View Embedding

The relation view characterizes the structure of KGs, in which entities are linked by relations. To preserve such relational structures, we adopt TransE [Bordes et al., 2013] to interpret a relation as a translation vector from its head entity to tail entity. Given a relation fact \( (h, r, t) \) in KGs, we use the following score function to measure the plausibility of the embeddings:

\[
f_{\text{rel}}(h^{(2)}, r, t^{(2)}) = -\|h^{(2)} + r - t^{(2)}\|_1
\]

(4)

where \( \| \| \) denotes either \( L_1 \) or \( L_2 \) vector norm. Then, we define the probability of \( (h, r, t) \) being a real relation fact (i.e., existing in the KGs) as follows:

\[
\mathcal{P}_\text{rel}(\zeta(h, r, t) = 1 | \Theta^{(2)}) = \text{sigmoid}(f_{\text{rel}}(h^{(2)}, r, t^{(2)})),
\]

(5)

where \( \Theta^{(2)} \) denotes the KG embeddings under the relation view and \( \zeta(h, r, t) \) denotes the label (“1” or “-1”) of \( (h, r, t) \). We parameterize \( \Theta^{(2)} \) by minimizing the following loss function below:

\[
\mathcal{L}(\Theta^{(2)} = \sum_{(h, r, t) \in \mathcal{X}^+ \cup \mathcal{X}^-} \log(1 + \exp(-\zeta(h, r, t)\cdot f_{\text{rel}}(h^{(2)}, r, t^{(2)}))),
\]

(6)

where \( \mathcal{X}^+ = \mathcal{X}_a \cup \mathcal{X}_b \) denotes the set of real relation facts in the source and target KGs, while \( \mathcal{X}^- \) denotes the set of faked ones sampled by replacing the head or tail entities of real relation facts with random ones.

3.5 Attribute View Embedding

For the attribute view, we use a convolutional neural network (CNN) to extract features from the attributes and values of entities. We splice the embeddings of an attribute \( a \) and its value \( v \) into a matrix, denoted by \( (a; v) \in \mathbb{R}^{2 \times d} \), and feed it to a CNN to obtain the compressed representation:

\[
\mathcal{C}_\text{NN}((a; v)) = \sigma(\text{vec}(\sigma((a; v) \ast \Omega))\mathbf{W}),
\]

(7)

where \( \mathcal{C}_\text{NN}(·) \) denotes a convolution operation that slides a convolution kernel \( \Omega \) of size \( 2 \times c \) over the input embedding matrix to extract high-level features. The feature map tensor is then reshaped to a vector by \( \text{vec}(·) \) and finally projected in the KG embedding space using a densely-connected layer parametrized by \( \mathbf{W} \). \( \sigma(·) \) is an activation function.

Given an attribute fact \( (h, a, v) \) in KGs, we define the following score function to measure its plausibility:

\[
f_{\text{attr}}(h^{(3)}, a, v) = -\|h^{(3)} - \mathcal{C}_\text{NN}((a; v))\|_1.
\]

(8)

Based on this, the head entities are expected to be close to its attributes and values. This objective can be achieved by minimizing the following loss function:

\[
\mathcal{L}(\Theta^{(3)} = \sum_{(h, a, v) \in \mathcal{Y}^{+}} \log(1 + \exp(-f_{\text{attr}}(h^{(3)}, a, v))),
\]

(9)

where \( \mathcal{Y}^{+} = \mathcal{Y}_a \cup \mathcal{Y}_b \) denotes the set of real attribute facts in the source and target KGs, and \( \Theta^{(3)} \) denotes the KG embeddings under the attribute view. Negative sampling is not employed here, because we find that it does not lead to noticeable improvement on entity alignment.

4 Cross-KG Training for Entity Alignment

In this section, we propose the cross-KG entity identity inference to capture the alignment information between two KGs, based on seed entity alignment. Furthermore, we present the cross-KG relation and attribute identity inference to enhance entity alignment, based on the automatically found relation and attribute alignment.

4.1 Entity Identity Inference

As shown in Figure 2, we take as an example the cross-KG entity identity inference under the relation view. Our intuition is that swapping aligned entities in their involved relation facts should lead to the same probability of identity inference, because the entities refer to the same object in the real world [Zhu et al., 2017; Sun et al., 2018]. Given a relation fact \( (h, r, t) \), if \( (h, \tilde{t}) \) appears in the seed entity alignment, we add the following auxiliary probability:

\[
\mathcal{P}_\text{rel}(\zeta(h, r, t) = 1 | \Theta^{(2)}) = \text{sigmoid}(f_{\text{rel}}(h^{(2)}, r, t^{(2)})).
\]

(10)

Also, if \( (t, \tilde{h}) \) exists in the seed entity alignment, we add:

\[
\mathcal{P}_\text{rel}(\zeta(h, r, t) = 1 | \Theta^{(2)}) = \text{sigmoid}(f_{\text{rel}}(h^{(2)}, r, t^{(2)})).
\]
We maximize these auxiliary probabilities over the relation facts having those entities in the seed entity alignment. The loss is computed as follows:

\[
\mathcal{L}_{CE}(\Theta^{(2)}) = \sum_{(h,r,t) \in \mathcal{X}'} \log(1 + \exp(-f_{rel}(\hat{h}^{(2)}, r, t^{(2)}))) + \sum_{(h,r,t) \in \mathcal{X}''} \log(1 + \exp(-f_{rel}(\hat{h}^{(2)}, r, t^{(2)}))),
\]

where \(\mathcal{X}'\) and \(\mathcal{X}''\) refer to the sets of relation facts whose head and tail entities are in the seed entity alignment, respectively.

For the cross-KG entity inference under the attribute view, the loss, denoted by \(\mathcal{L}_{CE}(\Theta^{(3)})\), is similarly defined:

\[
\mathcal{L}_{CE}(\Theta^{(3)}) = \sum_{(h,a,v) \in \mathcal{Y}'} \log(1 + \exp(-f_{attr}(\hat{h}^{(3)}, a, v))),
\]

where \(\mathcal{Y}'\) denotes the set of attribute facts such that the head entities are in the seed entity alignment.

### 4.2 Relation and Attribute Identity Inference

Similar to the cross-KG entity identity inference, we add the auxiliary probabilities for the cross-KG relation and attribute identity inference. Let us consider the relations for example. Given a relation fact \((h, r, t)\), if \((r, \hat{r})\) constitutes a relation alignment, we have:

\[
P_{rel}^{\alpha}(r, \hat{r}, t, t) = 1 | \Theta^{(2)} = \text{sigmoid}(f_{rel}(\hat{h}^{(2)}, r, t^{(2)})).
\]

Unlike the previous works [Zhu et al., 2017; Trsdale et al., 2019] that assume the existence of seed relation and attribute alignment, we propose a soft alignment method to automatically find the relation and attribute alignment in training. Here, “soft” means that we do not require the relations or attributes in the alignment to be strictly equivalent. In fact, due to the heterogeneity between KG schemata, finding relation and attribute alignment can be harder than entity alignment [Cheatham and Hitzler, 2014], and sometimes the strict alignment even does not exist. We denote the soft relation alignment by \(S_{rel}^{\alpha} = \{(r, \hat{r}, \hat{r}, \hat{r}) | | \text{sim}(r, \hat{r}) \geq \eta\}\), where \(r, \hat{r}\) are relations from different KGs, \(\text{sim}(r, \hat{r})\) denotes their similarity, and \(\eta\) denotes a threshold in \((0, 1]\). We consider two kinds of similarity: name similarity based on literal embeddings and semantic similarity based on relation embeddings. We combine them as a weighted sum:

\[
\text{sim}(r, \hat{r}) = \alpha_1 \cos(\phi(\text{name}(r)), \phi(\text{name}(\hat{r}))) + \alpha_2 \cos(\mathbf{r}, \mathbf{\hat{r}}),
\]

where \(\alpha_1, \alpha_2 > 0\) are two weighting factors and \(\alpha_1 + \alpha_2 = 1\). \(\cos(\cdot)\) calculates the cosine similarity of two embeddings.

We regard this similarity as a smooth coefficient to reduce the negative effect of inaccurate alignment and combine it into the loss of cross-KG relation identity inference:

\[
\mathcal{L}_{CRA}(\Theta^{(2)}) = \sum_{(h,r,t) \in \mathcal{X}''''} \log(1 + \exp(-f_{rel}(\hat{h}^{(2)}, r, t^{(2)}))),
\]

where \(\mathcal{X}''''\) denotes the set of relation facts having the relations in \(\mathcal{S}_{rel}^{\alpha}\). Note that, the soft relation alignment is not fixed but updated iteratively during the training process of MultiKE.

The cross-KG attribute identity inference can be formulated in the same way. Due to the space limitation, we omit its loss function \(\mathcal{L}_{CRA}(\Theta^{(3)})\) here.

### 5 View Combination

The view-specific embeddings characterize entity identities from different aspects. Intuitively, the general entity embeddings can benefit from multiple view-specific embeddings. In this section, we present three combination strategies.

#### 5.1 Weighted View Averaging

A straightforward combination is to average the embeddings from different views. To emphasize on important views, we assign weights to view-specific entity embeddings. Let \(\mathbf{h}\) denote the combined embedding for \(h\). Without loss of generality, let \(D\) be the number of views, and we have \(\mathbf{h} = \sum_{i=1}^{D} w_i \mathbf{h}^{(i)}\), where \(w_i\) is the weight of \(\mathbf{h}^{(i)}\), and can be calculated by:

\[
w_i = \frac{\cos(\mathbf{h}^{(i)}, \mathbf{\hat{h}})}{\sum_{j=1}^{D} \cos(\mathbf{h}^{(j)}, \mathbf{\hat{h}})},
\]

where \(\mathbf{\hat{h}}\) is the average of multi-view embeddings of \(h\), i.e., \(\mathbf{\hat{h}} = \frac{1}{D} \sum_{i=1}^{D} \mathbf{h}^{(i)}\). If the embedding from one view is far away from its average embedding, it would have a lower weight. This is a kind of late combination, because it aggregates embeddings after they have been learned independently.

#### 5.2 Shared Space Learning

This combination strategy seeks to induce an orthogonal mapping matrix from each view-specific embedding space to a shared space, based on the assumption that multiple views can be generated from the shared latent view. Let \(\mathbf{H}\) be the combined embedding matrix for all entities, where each row corresponds to an entity, and \(\mathbf{H}^{(i)}\) be the entity embedding matrix under the \(i\)th view. We minimize the mapping loss:

\[
\mathcal{L}_{SSL}(\mathbf{H}, \mathbf{Z}) = \sum_{i=1}^{D} (||\mathbf{H} - \mathbf{H}^{(i)}\mathbf{Z}^{(i)}||_F^2 + ||\mathbf{I} - \mathbf{Z}^{(i)\top}\mathbf{Z}^{(i)}||_F^2),
\]

where \(\mathbf{Z} = \bigcup_{i=1}^{D} \mathbf{Z}^{(i)}\). \(\mathbf{Z}^{(i)}\) serves as the mapping from the \(i\)th view-specific embedding space to the shared space, and \(|| \cdot ||_F\) denotes the Frobenius norm. The second term is used as a soft constraint to orthogonalize mapping matrices, where \(\mathbf{I}\) is the identity matrix. The orthogonality helps keep the distances between embeddings in the view-specific space unchanged during the transformation, and thus the shared space can learn the alignment information from the multiple embedding spaces. Note that this is also a kind of late combination.

#### 5.3 In-training Combination

Unlike the above strategies, this combination participates in the joint training of multi-view embeddings, which enables the multiple views to benefit from each other. The loss is:

\[
\mathcal{L}_{ITC}(\mathbf{H}, \mathbf{H}) = \sum_{i=1}^{D} (||\mathbf{H} - \mathbf{H}^{(i)}||_F^2),
\]

where \(\mathbf{H} = \bigcup_{i=1}^{D} \mathbf{H}^{(i)}\). The goal is to maximize the agreement between the combined embeddings and the view-specific embeddings in a unified embedding space.
Algorithm 1: Combined training process of MultiKE

Input: \( G_a, G_b \), word embeddings, max epochs \( Q \)

1. Train literal embeddings and get the name embeddings;
2. for \( q = 1, 2, \ldots, Q \) do
   3. Minimize \( L(\Theta(2)) \) under the relation view;
   4. Minimize \( L(\Theta(3)) \) under the attribute view;
   5. if in-training combination is used then
      6. Minimize \( L_{ITC}(H, H) \);
      7. Minimize \( L_{CE}(\Theta(2)) \) and \( L_{CE}(\Theta(3)) \);
      8. Update soft alignment \( S_{rel} \) and \( S_{attr} \);
      9. Minimize \( L_{CRA}(\Theta(2)) \) and \( L_{CRA}(\Theta(3)) \);
   10. if in-training combination is not used then
       11. Run weighted view averaging or shared space learning;
   12. Find entity alignment in \( H \) by nearest-neighbor search;

5.4 Combined Training Process

The complete training process is shown in Algorithm 1. The parameters are initialized with Xavier initializer and the loss functions are optimized with AdaGrad. We first train literal embeddings based on pre-train word embeddings [Mikolov et al., 2018] and character embeddings. Thus, we can directly obtain entity name embeddings. Then, we train (and combine, if in-training combination is used) embeddings from other views and perform the cross-KG entity, relation and attribute identity inference alternately. The soft alignment of relations and attributes is also updated iteratively during training. Finally, we combine the multiple view-specific entity embeddings if in-training combination is not used, and find entity alignment by the nearest-neighbor search. Note that, the name embeddings are retrieved from literal embeddings and they would not be updated in the follow-up training process.

6 Experiments

In this section, we report our experiments on MultiKE. The source code is accessible online.\(^3\)

6.1 Datasets

In our experiments, we reused two datasets, namely DBP-WD and DBP-YG, recently proposed in [Sun et al., 2018]. The two datasets were sampled from DBpedia, Wikidata and YAGO3, each of which contains 100 thousand aligned entity pairs, i.e., reference entity alignment. Each dataset provides 30% reference entity alignment as seed and leaves the remaining for evaluating entity alignment performance.

6.2 Comparative Methods

We compared MultiKE with seven recent embedding-based entity alignment methods, namely MTransE, IPTransE, JAPE, BootEA, KDCoE, GCN-Align and AttrE, which have been described in Section 2. We also extended TransD, HoE and ConvE for entity alignment on behalf of the three kinds of KG embedding models, because we found that their results were most competitive in our preliminary experiments. We merged the two KGs in a dataset as one by letting the two entities in each seed entity alignment have the same embedding and used these extended models to learn embeddings. We denote the MultiKE variants that employ weighted view averaging, shared space learning and in-training combination by MultiKE-WVA, MultiKE-SSL and MultiKE-ITC, respectively.

6.3 Experimental Settings

The following hyper-parameters were used in the experiments. Each training took \( Q = 200 \) epochs with learning rate 0.001. For the relation view embedding, 10 negative facts were sampled for each real relation fact. For the attribute view embedding, the number of filters was 2 and the convolution kernel size was \( 2 \times 4 \) (i.e., \( c = 4 \)). The activation function for the autoencoder and CNN was \( \text{tanh}(\cdot) \). For the relation and attribute identity inference, we set \( \alpha_1 = 0.6, \alpha_2 = 0.4 \) and \( \eta = 0.9 \). The embedding dimension \( d \) was set to 75 for all the comparative methods. We chose Hits@1 \( (k = 1, 10) \), mean rank (MR) and mean reciprocal rank (MRR) as the evaluation metrics. Higher Hits@\( k \) and MRR scores as well as lower MR scores indicate better performance. Note that, Hits@1 should be more preferable, and it is equivalent to precision widely-used in conventional entity alignment.

6.4 Entity Alignment Results and Analyses

Table 1 shows the comparison results of MultiKE and other embedding-based entity alignment methods. We found that MultiKE significantly outperforms the others on all the metrics across the two datasets. For example, on DBP-WD, MultiKE-SSL achieved a Hits@1 score of 91.86% with 17.07% absolute gain compared to BootEA (the second best method). This is because MultiKE explores the multi-view features of entity identities, while the others like MTransE, IPTransE and BootEA only learn from relation facts. As compared with AttrE, which leverages literals for entity alignment, MultiKE also achieved superior results (e.g., 52.9% improvement of Hits@1 on DBP-WD). This is due to the fact that the pretrained word embeddings in MultiKE can better capture the semantic similarity of literals than the character embeddings used in AttrE. Furthermore, AttrE embeds entities and literals in a unified space by TransE, which would fall short of handling the diversity of attribute values. Our three variants all achieved similar results. Note that the performance of MultiKE-SSL slightly decreased on DBP-YG. We think that this is because DBP-YG has stronger relation and attribute heterogeneity. For example, their relation numbers are 302 vs. 31, and the attribute numbers are 334 vs. 23, respectively. This hinders the combination by shared space learning.

**Effectiveness of views.** Table 2 depicts the entity alignment results based on view-specific entity embeddings when trained independently or by in-training combination. The three views all contributed to entity alignment, especially the name view. We owe it to the proposed literal embeddings, which can capture the semantic similarity of entity names. With in-training combination, the relation and attribute views benefited from the name view and also each other, thus their results improved a lot. As name embeddings are fixed, entity alignment results in the name view are the same in different combinations. This experiment indicated that entity names and word embeddings have great potentials for capturing the entity similarity.
Results under the relation and attribute views, with or without cross-KG relation or attribute identity inference.

**Effectiveness of cross-KG inference.** Here, we examined the effectiveness of the proposed relation and attribute identity inference. We additionally performed KG embedding under the relation and attribute views without optimizing $L_{\text{CRA}}(\Theta^{(2)})$ and $L_{\text{CRA}}(\Theta^{(3)})$, respectively. We denote such variants by “w/o CRA.” The entity alignment results under the relation and attribute views are shown in Figure 3. We observed that the relation and attribute identity inference brought improvement to entity alignment, especially in the attribute view. This is because the attribute heterogeneity is weaker than relation heterogeneity between different KGs. The soft attribute alignment was more accurate and thus contributed more.

**Analysis of Unsupervised Entity Alignment.** This task aims to align entities without seed entity alignment. As found in [Sun et al., 2017; Wang et al., 2018b], seed entity alignment is vital to learn KG embeddings from relation facts. We also encountered the same issue that the relation view would fall short if no seed entity alignment is given. However, as shown in Table 3, MultiKE-ITC still achieved acceptable results, thanks to the special in-training combination. The name view does not rely on seed alignment as supervision, while the relation and attribute views can benefit from it during training.

**Comparison with Conventional Methods.** We compared MultiKE with a famous and open-source conventional entity alignment method LogMap (version 2.4) [Jiménez-Ruiz et al., 2012]. In Table 4, we depict the results measured by the conventional precision, recall and F1-score. For embedding-based entity alignment, recall and F1-score are equal to precision, because the embedding-based methods always return a list of candidates for every input entity. LogMap is very competitive. In fact, it outperformed other comparative methods in Table 1. MultiKE achieved better results than LogMap, which again demonstrated its effectiveness and practicability.
References


[Qu et al., 2017] Meng Qu, Jian Tang, Jingbo Shang, Xiang Ren, Ming Zhang, and Jiawei Han. An attention-based collaboration framework for multi-view network representation learning. In CIKM, pages 1767–1776, 2017.


