Revisiting Embedding-based Entity Alignment: A Robust and Adaptive Method

Zequn Sun, Wei Hu*, Chengming Wang, Yuxin Wang, Yuzhong Qu

Abstract—Entity alignment—the discovery of identical entities across different knowledge graphs (KGs)—is a critical task in data fusion. In this paper, we revisit existing entity alignment methods in practical and challenging scenarios. Our empirical studies show that current work has a low level of robustness to long-tail entities and the lack of entity names or relation triples. We aim to develop a robust and adaptive entity alignment method, and the availability of relations, attributes, or names is not required. Our method consists of an attribute encoder and a relation encoder, representing an entity by aggregating its attributes or relational neighbors using the attention mechanisms that can highlight the useful attributes and relations in end-to-end learning. To let the encoders complement each other and produce a coherent representation space, we propose adaptive embedding fusion via a gating mechanism. We consider four evaluation settings, i.e., the conventional setting with both relation and attribute triples, as well as three challenging settings without attributes, without relations, without both relations and names, respectively. Results show that our method can achieve state-of-the-art performance. Even in the most challenging setting without relations and names, our method can still achieve promising results while existing methods fail.

Index Terms—knowledge graph embedding, entity alignment, adaptive embedding fusion.

1 Introduction

A knowledge graph (KG) is a repository of structured facts about the real world. Each fact is either a relation triple (*subject entity*, *relation*, *object entity*) or an attribute triple (*entity*, *attribute*, *value*). In recent years, many applications, such as question answering [14], recommender systems [41] and semantic search [48], incorporate KGs to help build their intelligence capabilities. However, as a KG is inherently incomplete, it frequently fails to provide sufficient knowledge to support downstream applications. Integrating various KGs for knowledge fusion and enrichment is an effective solution to this issue. Entity alignment, identifying equivalent entities across KGs, is a critical task in KG integration [30]. The root challenge for entity alignment stems from the heterogeneous symbolic representations of various KGs, such as different naming rules and multilingualism [30]. Recent advances in representation learning techniques hasten the advent of embedding-based entity alignment [2], [4], [5], [21], [31], [32], [34], [37], [44], [51], [55], [58]. It alleviates the heterogeneity problem by learning an embedding space to represent different KGs in which similar entities are kept close together while dissimilar ones are separated far apart. Entity similarities can thus be measured using the distances between entity embeddings. Existing entity alignment studies fall mainly into two broad categories, i.e., relation-based and attribute-enhanced methods. Relation-based entity alignment methods are built upon the assumption that similar entities usually have similar relational structures. They adopt structural learning techniques such as TransE [1] or graph neural networks (GNNs) like GCN [18], GAT [38] and RGCN [28] to capture the structure information of entities [2], [5], [23], [24], [32], [34], [42], [58]. The basic idea behind attribute-enhanced methods is to incorporate additional side information, such as attributes or attribute values, to improve embedding learning [4], [31], [37], [44], [55]. However, existing methods still have several severe limitations that weaken their robustness and practicability in real-world entity alignment scenarios.

Limitation 1. *Relation-based methods are unable to effectively handle the entities with few relation triples, let alone those without relations*. However, such entities typically account for a large proportion, resulting in the so-called long-tail issue [54]. We conduct a statistical analysis on the quantity distribution of entities in terms of the number of their attribute triples and relation triples in DBpedia [19]. Fig. 1 illustrates the statistics. More than half of entities (about 55%) have fewer than two relation triples and about 4% entities have no relations at all. This issue would cause inadequate embedding learning and therefore harm entity alignment performance [54]. Our empirical study in Section 3.1 indicates that existing relation-based methods such as MTransE [5], AliNet [34] and KEGCN [51] are actually very weak in long-tail entity alignment. To cope with long-tail entities, some literature pays particular attention to incorporating attribute triples [4], [36], [37], [44], [45], [46], [49], [51], [55] and exhibits improved performance. Attribute triples are a good complement for long-tail entities (refer to the first and second columns in Fig. 1). However, attribute-enhanced methods also suffer two limitations.

Limitation 2. *Most attribute-enhanced methods rely heavily on some discriminative attributes, especially name information, to compare entities or attributes. They also require manual attribute selection and value cleaning to pick out and process these attributes.* However, these methods [9], [25], [36], [44], [45], [46], [49], [52], [54], [55] would suffer from the unavailability of names or the issues caused by name abbreviations, aliases, or partial matches [43]. For instance, only 50.5% of entities in DBpedia

* *Corresponding author*

- *Zequn Sun, Wei Hu, Chengming Wang, Yuxin Wang and Yuzhong Qu are with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China.*
  E-mail: zqsun.nju@gmail.com, whu@nju.edu.cn, cmwang.nju@gmail.com, yuxinwangcs@outlook.com, yzqu@nju.edu.cn

have attribute *foaf:name*[1] to give name information. Furthermore, for cross-lingual entity alignment, the multilingualism makes it difficult to compare entities or attributes based on their names. Using translators to reconcile cross-lingual KGs is hampered by difficult cases [25]. See our empirical studies and findings in Section 3. Besides, although the discriminative attributes have excellent discriminative ability in identifying similar entities, different KGs usually use different URIs to denote these attributes, making it not a simple task to pick out these specific attribute URIs (e.g., those describing names). These methods usually require domain experts to go through the KG's schema definition (i.e., ontology). This may cost a significant amount of effort and time, as the ontology of a KG can be very complex, with multiple attributes for a specific meaning (see Table 2). When handling a new entity alignment task, we also need to create an entirely new attribute selection process from scratch. We also noticed that some methods like AttrE [37], IMUSE [12] and EMGCN [25] can make use of other general attributes, but they need hand-crafted rules for value cleaning to eliminate the heterogeneity. This limitation has received increasing attention in recent years [21], [35], [56], [57]. However, there has not been a solid solution to this issue yet. We believe that a robust method should not rely solely on one-side features such as names.

Limitation 3. *Incoherent representation spaces for relations and attributes hinder their mutual enhancement.* Relation triples and attribute triples are two different kinds of entity features. The embedding learning techniques for relation and attribute triples are also different. The graph structures are captured by GNNs in most relation-based methods [34], [42], [51], while the natural solution for representing literal values in attribute triples is to use pre-trained word embeddings or language models [36], [37], [55]. As a result of the orthogonal input features and different learning techniques, there is incoherence between the embeddings learned from relation triples and attribute triples. Existing attribute-enhanced methods directly merge the two embedding spaces via joint learning (e.g., AttrE [37]) or multi-view learning (e.g., MultiKE [55]). The resulted unified embedding space would suffer from the conflict between accurate cross-space transitions and embedding normalization, and thus cannot fully abstract the semantics in relation triples and attribute triples simultaneously [6]. This challenge even prevents some attribute-enhanced methods from representing attribute triples. Consequently, they leverage attribute triples in an offline way. For example, JAPE [31] abandons the use of attribute values. IMUSE [12] and EMGCN [25] compute an attribute-based similarity for entities by comparing the string similarity of attributes and values. The attribute-based similarity is finally combined with the relation-based embedding similarity for ensemble alignment retrieval. We argue that this offline method cannot fully exploit the complementarity of relation and attribute triples and fail to let them enhance each other.

To resolve the above limitations, we propose RoadEA, a robust and adaptive entity alignment method. It employs an attribute encoder and a relation encoder to learn from both attribute and relation triples in an adaptive manner

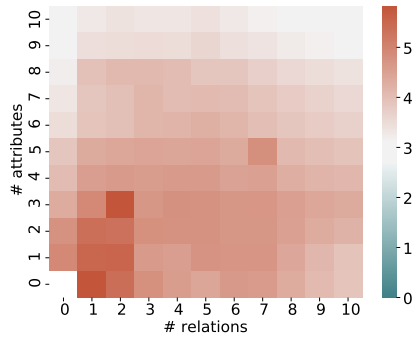


Fig. 1: Quantity distribution of entities in terms of the number (in $\log 10$ scale) of attribute and relation triples in DBpedia.[2] Darker red indicates a larger number. The value at $(0, 0)$ is omitted since an entity has at least one relation or attribute triple. Due to space limitation, we only show the distribution where the number of relations or attributes is less than 10.

(*to address Limitation 1*). The attribute encoder seeks to take full advantage of general attributes rather than focusing only on entity names. We employ a pre-trained language model, e.g., BERT [8], to generate the representations of attribute values. We do not use value cleaning strategies and assume that BERT can handle the symbolic heterogeneity and output similar embeddings for semantically related values (*to address Limitation 2*). To represent an attribute, we consider its correlated attributes and value representations rather than its name. We first build a weighted attribute association graph and initialize the embedding of an attribute by performing a 1D convolution operation over its value representations. Then, we apply a graph convolution operation to the graph to obtain attribute embeddings. To represent an entity, we use the attention mechanism to aggregate the representations of its attribute-value pairs. We have no special treatment of entity names, and the attention mechanism can highlight helpful attributes in the learning process without manual attribute selection (*to address Limitation 2*). In the relation encoder, an entity is represented by aggregating its relational neighbors with an attention mechanism. We use a gate mechanism to stack the two encoders for adaptive embedding fusion, allowing for adequate interactions and mutual enhancement (*to address Limitation 3*). We feed the output embeddings of the attribute encoder as the initial entity representations in the relation encoder to jump-start the layer-by-layer aggregation. The two encoders are loosely coupled. If an entity has no attributes, its embedding is randomly initialized and trained using the relation encoder. If an entity has no relations, its attribute-aware embedding is the final representation. The output representations of adaptive embedding fusion are used for alignment learning. Our method does not require an entity to possess attribute or relation triples. The gate mechanism to stack the two encoders can learn a unified representation space, avoiding the incoherent representations for relations and attributes.

Our main contributions are summarized as follows:

- We revisit existing embedding-based entity alignment methods by analyzing their limitations and investigating

1. http://xmlns.com/foaf/0.1/name is the uniform resource identifier (URI) of an attribute that describes entity names.

2. We use the mapping-based triples of English DBpedia, which can be downloaded from http://downloads.dbpedia.org/2016-10/core/.

their robustness to challenging entity alignment settings. Empirical studies indicate that relation-based methods cannot handle long-tail entities, and attribute-enhanced methods heavily rely on entity names to provide alignment signals. As a result, they would suffer from the unavailability of such information (e.g., relation triples or entity names) that they need in real scenarios, leading to a low level of robustness and adaptability. Our work calls for more attention to robust entity alignment.

- We propose RoadEA, a robust and adaptive embedding method for entity alignment. It stacks an attribute encoder and a relation encoder with a gate mechanism for adaptive embedding fusion. Its biggest advantage lies in the good robustness to the unavailability of relation triples, attribute triples or name information. To the best of our knowledge, our work is the first attempt to tackle these challenging settings of entity alignment.
- We consider four practical settings to evaluate RoadEA, i.e., entity alignment (i) with attributes and relations, (ii) without attributes, (iii) without relations, and (iv) without relations and names. Experimental results on the OpenEA dataset [35] demonstrate the effectiveness and robustness of RoadEA. Even in the most challenging setting (i.e., the fourth one), RoadEA can still achieve promising performance while existing methods fail.

The remainder of this paper proceeds as follows. We first introduce necessary preliminaries and review related work in Section 2. Then in Section 3, we report our empirical studies and findings. In Section 4, we describe the proposed method. We report experimental settings and results in Section 5. Finally, we conclude the paper with future work in Section 6.

2 Preliminaries and Related Work

2.1 Definitions and Notations

We introduce the definitions and notations used in this paper.

Knowledge graph. We define a KG as a six-tuple, i.e., $\mathcal{K} = \{\mathcal{E}, \mathcal{R}, \mathcal{A}, \mathcal{V}, \mathcal{T}_{\text{rel}}, \mathcal{T}_{\text{att}}\}$. $\mathcal{E}$ and $\mathcal{R}$ denote the sets of entities and relations, respectively. $\mathcal{A}$ is the set of attributes and $\mathcal{V}$ is the set of attribute values. $\mathcal{T}_{\text{rel}} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ is the set of relation triples. $\mathcal{T}_{\text{att}} \subseteq \mathcal{E} \times \mathcal{A} \times \mathcal{V}$ is the set of attribute triples.

Relation. A relation connects two entities. For example, the relation triple (*Kobe Bryant*, *birthplace*, *Philadelphia*) in DBpedia indicates that there is a relationship *birthplace* between the two entities *Kobe Bryant* and *Philadelphia*. Although a KG is typically built around relation triples, there are still some entities that do not have any relations as illustrated by Fig. 1. Existing studies pay little attention to these entities.

Attribute. Attributes give the inherent properties of entities, and attribute values are literals. For example, the attribute triple (*Kobe Bryant*, *birthDate*, "*1978-08-23*") gives the birthday of Kobe Bryant. Attributes are essential features of entities and can complete entities' information as indicated by Fig. 1.

Entity alignment. Given a source KG $\mathcal{K}_1$ and a target KG $\mathcal{K}_2$, entity alignment is the task of identifying the one-to-one correspondences between their entities $\mathcal{E}_1$ and $\mathcal{E}_2$, i.e., $\mathcal{D} = \{(e_1, e_2) \in \mathcal{E}_1 \times \mathcal{E}_2 \,|\, e_1 \equiv e_2\}$, where "$\equiv$" denotes the equivalence relationship. In most cases, a small set of pre-aligned entity pairs $\mathcal{D}_{\text{train}} \subset \mathcal{D}$ is provided as training data to jump-start the embedding and alignment learning.

Alignment retrieval and evaluation metrics. Given a source entity, an embedding-based method ranks the entities of the target KG in descending order based on their embedding similarities to the source entity. The ground-truth counterpart is expected to be at the first position. The widely used metrics to assess the performance are H@k (e.g., $k = 1, 5$) and MRR (mean reciprocal rank). H@k measures the percentage of the test entity pairs whose target entities are ranked in top k, i.e.,

$$\text{H@}k = \frac{|\{(e_1, e_2) \in \mathcal{D}_{\text{test}} \,|\, \text{Rank}_{e_2} \leq k\}|}{|\mathcal{D}_{\text{test}}|}, \tag{1}$$

where $\mathcal{D}_{\text{test}}$ denotes the test entity alignment. Rank_{e_2} is the rank of the correct target counterpart in the sorted candidate list. The target counterpart entity is expected to be ranked at the top. MRR is the average of the reciprocal ranks:

$$\text{MRR} = \Big(\sum_{(e_1, e_2) \in \mathcal{D}_{\text{test}}} \frac{1}{\text{Rank}_{e_2}} \Big) \cdot \frac{1}{|\mathcal{D}_{\text{test}}|}. \tag{2}$$

Higher H@k and MRR scores indicate better performance.

Notations. We use boldface lowercase and uppercase letters to denote vectors (e.g., embeddings) and matrices, respectively. For example, $\mathbf{e}_1$ and $\mathbf{e}_2$ denote the embeddings of entities e_1 and e_2, respectively. $\mathbf{W}$ denotes a weight matrix.

Datasets. We use the OpenEA dataset [35] in our work. It has four entity alignment settings, i.e., EN-FR and EN-DE (extracted from the multilingual DBpedia), as well as D-W (DBpedia-Wikidata), and D-Y (DBpedia-YAGO). Each setting has two scales with $15,000$ (15K) and $100,000$ (100K) entity alignment pairs, respectively. Please refer to Section 5.2 for more details. We abandon D-Y due to the name bias issue that almost all the aligned entities in DBpedia and YAGO have the same name. Hence, D-Y is unsuitable for assessing the actual performance and robustness of attribute-enhanced methods [21], [56], [57]. We do not use DBP15K [31] because it only provides entity attributes without values and is denser than real KGs, which is insufficient for our problem setting. We also do not choose DWY100K [32], which was originally designed for relation-based entity alignment. By contrast, OpenEA can provide realistic evaluation settings.

2.2 Related Work

2.2.1 Relation-based Entity Alignment

Relation triples establish the relational topological structures of entities. A widely-adopted assumption for relation-based entity alignment is that the entities with similar relational structures should have similar embeddings. Two different relational learning techniques are widely used for entity embedding learning. One is the translational KG embedding technique such as TransE [1]. It is adopted by many methods including JAPE [31], MTransE [5], IPTransE [58], BootEA [32], SEA [26] and TransEdge [33]. The other technique is graph neural networks (GNNs), including GCN [18], GAT [38] and RGCN [28], which have drawn extensive attention in recent years. Many studies present their GNN variants for entity alignment, such as the vanilla GCN in GCNAlign [42], the multi-channel GNN in MuGNN [2], the relation-aware GNNs in RDGCN [44], HGCN [45], MRAEA [23] and KEGCN [51], the multi-hop GNN in AliNet [34], as well as the multi-order GNN in EMGCN [25]. All these methods require the availability of relation triples. By contrast, our method does not rely on relation triples.

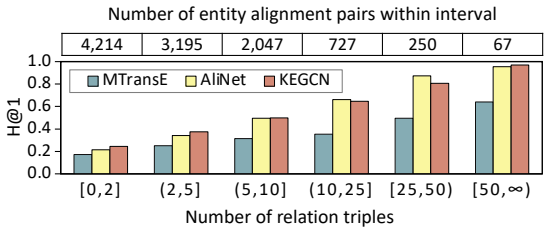


Fig. 2: H@1 w.r.t. the number of rel. triples on EN-FR-15K.

2.2.2 Attribute-enhanced Entity Alignment

Although attribute-enhanced methods also have a relational learning model to learn from relation triples, their key idea is to incorporate side information to improve embedding learning. JAPE [31] and GCNAlign [42] consider attribute correlations to cluster similar entities. They assume that the entities whose attributes are highly correlated (e.g., *latitude* and *longitude*) should be aligned with high probability. However, they ignore attribute values, and attribute correlations cannot provide an accurate comparison to identify aligned entities. Due to the high symbolic heterogeneity in attribute values, most of the follow-up studies choose to skirt around this challenge. They select some simple but discriminative attribute values such as entity names [9], [25], [36], [50], [53], [55], descriptions [4] and images [20] to enhance embedding learning, and achieve promising performance. Inspired by this, many recent GNN-based methods such as RDGCN [44], HGCN [45] and GMNN [49] use entity name embeddings to initialize their input layers, which has almost become the *de facto* initialization method for GNN-based entity alignment. These methods are called name-enhanced methods in this paper. However, we find that their performance would drop a lot without name embeddings (please see our empirical study in Section 3). Besides, as discussed in Section 1, these methods may also suffer from the unavailability of the discriminative attributes in real-world scenarios. MultiKE [55] and AttrE [37] can learn from general attribute triples. Both of them regard the attribute values as "nodes" (and attributes as "relations") such that the relational learning techniques like TransE [1] and ConvE [7] can be used again for embedding learning. We find that their success is also attributed to the use of name information (please see the results in Table 3). IMUSE [12] and EMGCN [25] can also utilize general attribute triples, but they need hand-craft rules for value cleaning to compare attribute values. They also rely on machine translation to reconcile cross-lingual KGs. Our method does not rely on the name information, value cleaning or machine translation.

3 Empirical Study

In this section, we report our empirical studies and seek to answer the following research questions:

3.1 What about the Performance on Long-tail Entities?

It is intuitive that aligning entities with rich relation triples is easier than long-tail ones. In this study, we compare the performance of relation-based methods on aligning rich and long-tail entities. We divide the aligned entity pairs in the test data of EN-FR-15K into several groups based on the number of their relation triples. The triple number of a test entity pair is defined as the average number of relation triples that the involved entities have. The performance is assessed by H@1. Fig. 2 compares the results of three popular relation-based methods: MTransE [5], AliNet [34] and KEGCN [51]. They all perform poorly in aligning long-tail entities (those with fewer than 2 relation triples), and their H@1 scores increase as the number of relation triples increases. This is in accord with our expectations, as more relation triples provide more features for embedding learning. The finding demonstrates the vulnerability of relation-based methods. It also reveals the importance of incorporating attribute triples.

TABLE 1: Number of aligned attribute triples in OpenEA.

	EN-FR-15K	EN-DE-15K	D-W-15K
# aligned entity pairs	15,000	15,000	15,000
# aligned name triples	8,710	12,068	2,222
# aligned attribute triples	27,649	28,887	13,895
# aligned attribute triples w/o names	18,135	16,360	8,393

3.2 What if Entity Names Are not Available?

We studied the OpenEA dataset [35] to see whether aligned entities have similar names. Non-English attributes and values are translated into English. Table 1 shows the number of aligned name triples[3] of aligned entities. The attributes used in name triples are given by Table 2. We find a large number of aligned entity pairs whose two entities have the same name. This explains why many methods use name embeddings to assist entity alignment. However, these methods suffer from a sharp performance reduction when lacking names. We compare the popular name-enhanced methods AttrE [37], MultiKE [55] and RDGCN [44] in two settings with or without entity names, respectively, as shown in Table 3. Refer to Section 5.5.2 for implementation details. The H@1 of RDGCN declines from 0.755 to 0.255 when entity names are unavailable. This finding provides a new insight of existing name-enhanced entity alignment methods and calls for robust solutions. Besides, although each entity is endowed with a literal name, in the case of the low-resource setting or when entity names are too heterogeneous, entity names may not provide helpful alignment signals. This finding is supported by the small number of aligned name triples in D-W-15K. The heterogeneity in DBpedia and Wikidata is more significant than that in EN-FR and EN-DE. Fortunately, there are also a larger number of aligned attribute triples except for name triples. Our work seeks to make use of these general attribute triples.

3.3 Is Manual Feature Selection Really Robust?

We agree that some manually selected attributes, such as entity names and descriptions, can help entity alignment. However, due to the schema heterogeneity of different KGs, it is a non-trivial task to retrieve entity names from attribute triples. For instance, after reviewing the attributes in DBpedia [19], Wikidata [39] and YAGO3 [27], we find (at least) 11 attributes used to describe entity names, as listed in Table 2. Except for the widely-used core ontologies of the Semantic Web, e.g., the RDF Schema,[4] SKOS Schema[5] and FOAF

3. The attribute triples describing entity names are called name triples, e.g., (*dbr:Kobe_Bryant*, *foaf:name*, "*Kobe Bean Bryant*") in DBpedia.
4. https://www.w3.org/TR/rdf-schema/
5. https://www.w3.org/TR/swbp-skos-core-spec/

TABLE 2: Attributes used to describe entity names.

Sources	Attribute URIs
RDF Schema	http://www.w3.org/2000/01/rdf-schema#label
SKOS	http://www.w3.org/2004/02/skos/core#altLabel
FOAF	http://xmlns.com/foaf/0.1/name http://xmlns.com/foaf/0.1/givenName http://xmlns.com/foaf/0.1/nick
DBpedia	http://dbpedia.org/ontology/birthName http://dbpedia.org/ontology/alias http://dbpedia.org/ontology/longName http://dbpedia.org/ontology/otherName http://dbpedia.org/ontology/pseudonym
Wikidata	http://www.wikidata.org/entity/P373

TABLE 3: Entity alignment results w/ or w/o name triples.

Methods	EN-FR-15K		EN-DE-15K		D-W-15K	
	H@1	MRR	H@1	MRR	H@1	MRR
AttrE [37]	0.481	0.569	0.517	0.597	0.299	0.381
AttrE (w/o names)	0.234	0.327	0.310	0.404	0.229	0.312
MultiKE [55]	0.749	0.782	0.756	0.782	0.411	0.468
MultiKE (w/o names)	0.349	0.458	0.421	0.522	0.319	0.414
RDGCN [44]	0.755	0.800	0.830	0.859	0.515	0.584
RDGCN (w/o names)	0.255	0.355	0.511	0.592	0.331	0.409

Vocabulary,[6] DBpedia and Wikidata both define their own attributes to describe the name information of entities. In this case, the selected attributes in Table 2 may be inapplicable to other KGs that use different schemata for extracting entity names. When confronted with KGs that do not completely reuse the core ontologies of the Semantic Web (e.g., DBpedia) or the URIs are not in human-readable format (e.g., Wikidata), it would cost a commitment of time and energy to pick out helpful attributes. As a result, name-enhanced methods have low robustness and adaptability. Differently, our work seeks to remove the dependency on manual feature selection.

3.4 How Many Alignment Signals Can Attributes Offer?

Attribute triples are an essential ingredient of a KG. For instance, the English DBpedia has $14,388,539$ attribute triples. The number is very close to that of relation triples, which is $18,746,176$. We think that it is unwise to abandon attribute triples in embedding learning. However, given the lack of attribute alignment and the heterogeneity of literal values, it remains to be seen whether the general attribute triples (aside from name triples) contain useful information for identifying entities. To find an answer, we have looked into the OpenEA dataset. Based on the manually checked attribute alignment (found by string matching and machine translation), e.g., *birthday* and *geburtstag*, as well as the value alignment found by exact string matching, we find a large number of aligned attribute triples as reported in Table 1. For example, in EN-FR-15K, there are $27,649$ aligned attribute triples, and more than 82.4% of aligned entity pairs have at least one aligned triple. Except for aligned name triples, there are still $18,135$ aligned attribute triples, suggesting that general attribute triples can also offer alignment clues.

4 Robust and Adaptive Entity Alignment

Our framework is depicted in Fig. 3. The attribute encoder encompasses value embedding, attribute embedding, and attribute-aware entity embedding. The relation encoder embeds entities by aggregating relational neighbors. The two encoders are adaptively combined for embedding fusion.

6. http://xmlns.com/foaf/spec/

4.1 Value Embedding

We represent the literal values in attribute triples by looking up pre-trained language models or word embeddings. Given a literal value v consisting of n tokens, denoted by $v = (t_1, t_2, \ldots, t_n)$, the tokens $t_1, t_2, \ldots, t_n$ are first encoded by pre-trained language models or word embeddings:

$$\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_n = \text{Encoder}(t_1, t_2, \ldots, t_n), \tag{3}$$

where Encoder() can be the BERT [8] or fastText [16] encoder that takes a sequence of tokens as input and outputs a sequence of the corresponding token embeddings. Then, the value representation $\mathbf{v}$ is calculated as

$$\mathbf{v} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{t}_i. \tag{4}$$

Value embeddings are used for attribute embedding, and the encoder itself does not participate in training. If Encoder() has a tokenizer, we use it to preprocess input tokens, such as removing stop words and special characters, to avoid unexpected tokenization and ensure an effective lookup. We do not employ additional preprocessing strategies to clean or reformat values such that our method can be applied to different KGs without requiring special customization. We discover that BERT can deal with the symbolic heterogeneity in literal values and generate similar embeddings for semantically related values. It can also handle numeric values.

4.2 Attribute Embedding

Attribute embedding is a non-trivial task because attribute names may not deliver clear semantic meanings and there would be multiple attributes with different names referring to the same meaning (as shown in Table 2). For example, the name of attribute http://www.wikidata.org/entity/P373 is "commons category", but we find that it is usually used to describe the entity name information in Wikidata. We think that the semantic information of an attribute can be reflected by its values and correlated attributes in the KG. Specifically, given an attribute a, we first extract all its values appeared in the KG, denoted by $(v_1^a, v_2^a, \ldots, v_m^a)$, where m is the number of values. Then, we build a 1D convolution layer Conv() over the corresponding value embeddings $[\mathbf{v}_1^a, \mathbf{v}_2^a, \ldots, \mathbf{v}_m^a] \in \mathbb{R}^{m\times d_{\text{word}}}$ to get the combined representation of values $\text{Conv}([\mathbf{v}_1^a, \mathbf{v}_2^a, \ldots, \mathbf{v}_m^a]) \in \mathbb{R}^{m\times 1}$, where d_{word} is the dimension of word embeddings. Each kernel outputs a representation for these values. Let c denote the number of convolution kernels in the layer. We use mean pooling to merge the c representations and feed the output into a linear layer to obtain the final representation:

$$\mathbf{a}_{\text{conv}} = \mathbf{W}_{\text{conv}}\Big(\frac{1}{c}\sum_{i=1}^{c}\text{Conv}_i([\mathbf{v}_1^a, \mathbf{v}_2^a, \ldots, \mathbf{v}_m^a])\Big), \tag{5}$$

where $\mathbf{W}_{\text{conv}} \in \mathbb{R}^{d_{\text{val}}\times m}$ represents the weight matrix in the linear layer. $\text{Conv}_i()$ denotes the i-th 1D convolution kernel used to learn high-level representations from value embeddings while reducing output dimension. Our method

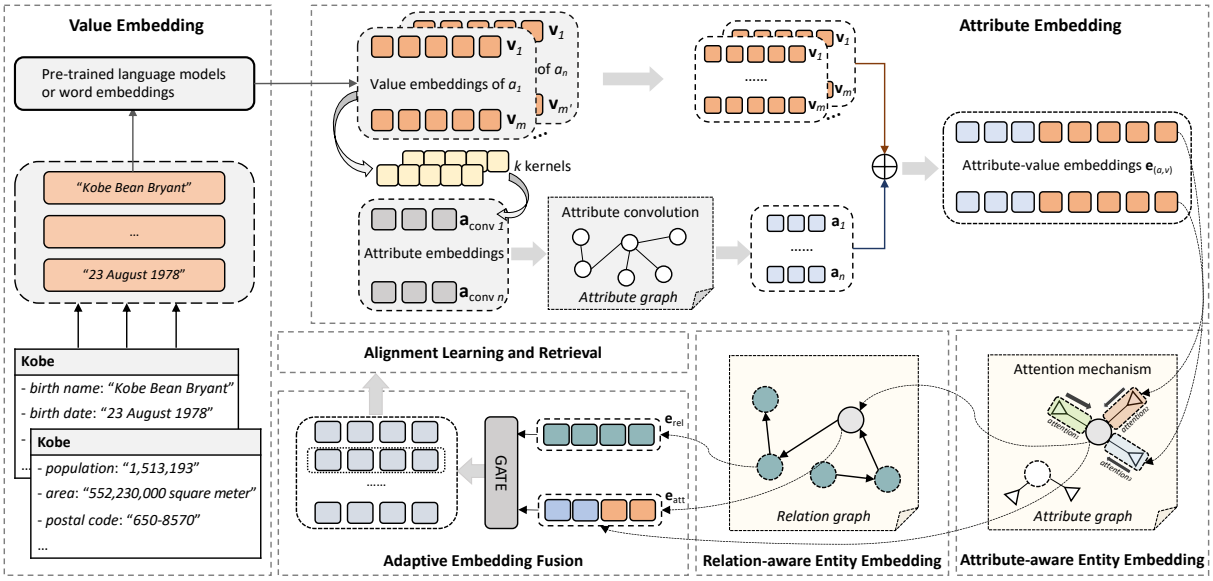


Fig. 3: Overview of RoadEA. *Value embedding* generates representations for literal values with a pre-trained language model. *Attribute embedding* uses convolution techniques to learn attribute embeddings from value representations and the attribute association graph. *Attribute-aware entity embedding* uses attribute-value pair embeddings to represent entities by the attention mechanism. *Relation-aware entity embedding* aggregates the representations of relational neighbors. *Adaptive embedding fusion* combines attribute-aware and relation-aware embeddings through a gate mechanism for *alignment learning and retrieval*.

is invariant to the order of input embeddings, and does not require attributes to have the same number of values.

In addition to learning from attribute values, we think that the correlated attributes should also participate in the embedding learning of each other. This is because the correlated attributes, e.g., *latitude* and *longitude*, typically express relevant semantics. Different from JAPE that captures pairwise attribute correlation [31], our method models such correlation in a global way to fully exploit the high-order relevance between attributes. We first construct an attribute association graph for each KG, where each attribute appears as a node and the edge between two attributes is assigned a weight to indicate the co-occurrence frequency, as shown in Fig. 4. Specifically, if two attributes are used together to describe an entity, we then add an edge between the two attributes, or increase the weight of their edge by 1 if the edge already exists. To leverage the information of the attribute itself, we add a self-loop edge for each attribute. The weights are finally normalized as

$$\omega(a_i, a_j) = \frac{2 \cdot \text{occu}(a_i, a_j)}{\sum_{a \in \mathcal{A}} (\text{occu}(a_i, a) + \text{occu}(a_j, a))}, \tag{6}$$

where $\omega(a_i, a_j)$ denotes the normalized edge weight between attributes a_i and a_j, and the function $\text{occu}(a_i, a_j)$ counts the occurrences (i.e., edge weights) of the two attributes.

We use GCN [18] to learn attribute embeddings by propagating over the association graph. The output representation matrix at the $(l+1)$-th layer, denoted as $\mathbf{A}^{(l+1)}$, is

$$\mathbf{A}^{(l+1)} = \sigma(\widetilde{\mathbf{D}}^{-\frac{1}{2}} \widetilde{\mathbf{H}} \widetilde{\mathbf{D}}^{-\frac{1}{2}} \mathbf{A}^{(l)} \mathbf{W}_{\text{asso}}^{(l+1)}), \tag{7}$$

where $\widetilde{\mathbf{H}}$ denotes the adjacency matrix of the weighted association graph and $\widetilde{\mathbf{D}}_{ii} = \sum_j \widetilde{\mathbf{H}}_{ij}$. $\mathbf{A}^{(l)}$ is the attribute representation matrix of the l-th layer. $\mathbf{W}_{\text{asso}}^{(l+1)}$ is a weight matrix in the $(l+1)$-th GCN layer. The input representation of attribute a at the first layer is its value-based embedding $\mathbf{a}_{\text{conv}}$. σ denotes the sigmoid function. We use the output representations of the GCN as the final attribute embeddings, e.g., denoted as $\mathbf{a}$ for attribute a.

4.3 Attribute-aware Entity Embedding

After obtaining value and attribute embeddings, given an attribute triple (e, a, v), we encode the attribute-value pair (a, v) to represent entity e as follows:

$$\mathbf{e}_{(a,v)} = \text{norm}(\mathbf{a} \,||\, \mathbf{v}), \tag{8}$$

where $\text{norm}()$ denotes the L_2 normalization for reducing the trivial optimization procedure of artificially increasing vector norm [1], and $||$ denotes concatenation. In this way, the attribute-aware embedding of an entity can be learned by inductively aggregating all its attribute-value pairs. Given that not all attribute triples are helpful in identifying entities, we use the graph attention mechanism in GAT [38] to learn weights for different attributes in an end-to-end manner. As a result, our method does not need the aforehand manual selection to find discriminative attributes. For an entity e, we use the mean representation of its attribute-value pairs as its initialization, denoted as $\mathbf{e}$, to jump-start the calculation of the initial attention weights. The weights and attribute-aware entity embeddings are further updated in the subsequent training. Specifically, the weight between entity e and its attribute-value pair (a, v) is calculated as follows:

$$\alpha_{(e,a,v)} = \frac{\exp\left(\tau(\mathbf{v}^\top [\mathbf{W}_{\text{att}}\mathbf{e} \,||\, \mathbf{W}_{\text{att}}\mathbf{e}_{(a,v)}])\right)}{\sum_{(e,a',v') \in \mathcal{T}_{\text{att}}^{(e)}} \exp\left(\tau(\mathbf{v}^\top [\mathbf{W}_{\text{att}}\mathbf{e} \,||\, \mathbf{W}_{\text{att}}\mathbf{e}_{(a',v')}])\right)}, \tag{9}$$

where $\mathbf{W}_{\text{att}}$ is the weight matrix of a single-layer feed forward network for embedding transformation, $\mathcal{T}_{\text{att}}^{(e)}$ represents the set of attribute triples of entity e. $\tau()$ denotes the leaky rectified linear unit [22]. We update e's attribute-aware entity

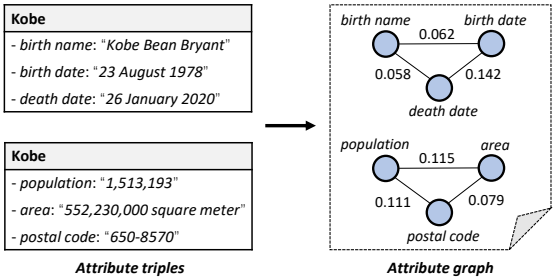


Fig. 4: Illustration of attribute graph construction in DBpedia.

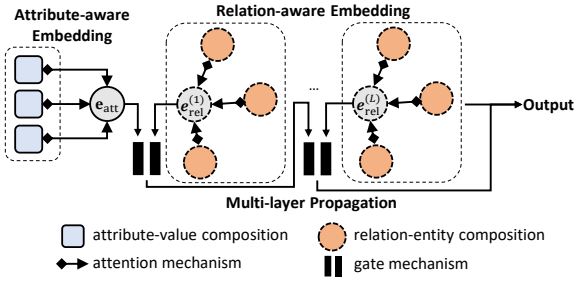


Fig. 5: Adaptive embedding fusion.

embedding by aggregating its attribute triple embeddings using the weights as follows:

$$\mathbf{e}_{\text{att}} = \text{norm}\Big(\sum_{(e,a,v)\in\mathcal{T}_{\text{att}}^{(e)}} \alpha_{(e,a,v)}\mathbf{e}_{(a,v)} + \mathbf{e}\Big). \quad (10)$$

The output attribute-aware entity embeddings can be directly used for alignment learning and retrieval. They can also be combined with relation-aware entity representations for adaptive embedding learning.

4.4 Relation-aware Entity Embedding

To take advantage of relational structures for entity alignment, we further propose the relation encoder to aggregate and propagate entity embeddings over relational neighborhood subgraphs. In this way, our method can also be applied to relation-based entity alignment. Relation triples establish the various connections among entities. We think that different relation triples have different effects on discriminating entities. Besides, relation types can also reflect the importance of relational neighbors. Therefore, we consider the types of relations when updating entity representations over relational subgraphs. Different from R-GCN [28] that distinguishes between relational neighbors in a single KG, in our problem setting, we also take into consideration the impact of heterogeneous relations between different KGs. We define the following equation to calculate the weight between entity e and its relation r based on the l-th layer:

$$\beta_{(e,r)}^{(l)} = \frac{\exp(\mathbf{e}_{\text{rel}}^{(l)} \cdot \mathbf{r})}{\sum_{(e,r',e')\in\mathcal{T}_{\text{rel}}^{(e)}} \exp(\mathbf{e}_{\text{rel}}^{(l)} \cdot \mathbf{r}')}, \quad (11)$$

where $\mathcal{T}_{\text{rel}}^{(e)}$ is the set of relation triples of entity e. $\mathbf{e}_{\text{rel}}^{(l)}$ denotes the relation-aware embedding of entity e at the l-th layer. Many existing neighborhood aggregation schemes [34] do not capture the relation information. By contrast, we concatenate the relation embedding and the corresponding neighbor embedding as the relation-neighbor pair representation for aggregation. Specifically, the representation of entity e at the $(l+1)$-th layer is calculated as follows:

$$\mathbf{e}_{\text{rel}}^{(l+1)} = \text{norm}\Big(\mathbf{W}_{\text{rel}}^{(l+1)} \sum_{(e,r,e')\in\mathcal{T}_{\text{rel}}^{(e)}} \beta_{(e,r)}^{(l)} (\mathbf{r} \,||\, \mathbf{e}'_{\text{rel}}{}^{(l)})\Big), \quad (12)$$

where matrix $\mathbf{W}_{\text{rel}}^{(l+1)}$ is for embedding transformation. At the beginning, an entity has no initial representation, i.e., we do not know $\mathbf{e}_{\text{rel}}^{(0)}$ and $\mathbf{e}'_{\text{rel}}{}^{(0)}$. The most common strategy is to generate initial representations randomly [1], although it may increase the difficulty and instability of network training. Another solution is to initialize the representations with name embeddings [44], which is not robust as we have discussed in Section 3, so we abandon it. The relation encoder can also work independently for entity embedding given randomly generated initial representations. Note that the relation-aware embedding does not contain the information about the central entity itself, leaving an interface for the adaptive fusion of the attribute and relation encoders.

4.5 Adaptive Embedding Fusion

As mentioned in Section 1, combining relation-based and attribute-based embeddings remains a challenge due to their independent representation spaces. We combine the two encoders sequentially to produce a unified entity embedding space through layer-by-layer training. In detail, we use the attribute-aware embedding of an entity e as its initial representation in the relation encoder, which is further combined with its relation-aware embedding for propagation. If the entity has no attributes, its initial representation is randomly generated. We have

$$\mathbf{e}_{\text{rel}}^{(0)} = \begin{cases} \mathbf{e}_{\text{att}} & \text{if entity } e \text{ has attribute triples} \\ \mathbf{e}_{\text{rand}} & \text{if entity } e \text{ has no attribute triples} \end{cases}. \quad (13)$$

It should be noted that the relation and attribute encoders can learn embeddings independently. Hence, our method does not rely on the existence of an entity's relation or attribute triples. Furthermore, we do not assume that attribute and relation triples contribute equally to embedding learning. Inspired by the skipping connections in neural networks [29], we use the gate mechanism to combine the entity's representation itself with its relation-aware embedding. The gate mechanism allows helpful information to enter the following layers. Fig. 5 illustrates the proposed adaptive embedding fusion. Specifically, at the $(l+1)$-layer's neighborhood aggregation ($l \geq 0$), the entity is represented as

$$\mathbf{e}^{(l+1)} = \text{G}(\mathbf{e}_{\text{rel}}^{(l)})\, \mathbf{e}^{(l)} + \big(1 - \text{G}(\mathbf{e}_{\text{rel}}^{(l)})\big)\, \mathbf{e}_{\text{rel}}^{(l)}, \quad (14)$$

where $\text{G}(\mathbf{e}_{\text{rel}}^{(l)}) = \sigma(\mathbf{W}\mathbf{e}_{\text{rel}}^{(l)} + \mathbf{b})$ serves as the gate to control the embedding fusion of the entity and its neighbors.

To further strengthen the respective embeddings learned from attribute triples and relation triples, at the output layer, we concatenate the attribute-aware and relation-aware embedding of an entity as its final representation:

$$\hat{\mathbf{e}} = \mathbf{e}_{\text{att}} \,||\, \mathbf{e}_{\text{rel}}^{(L)}, \quad (15)$$

where L is the number of layers for neighbor aggregation. We have $\mathbf{e}_{\text{att}} \in \mathbb{R}^{d_{\text{att}}}$, $\mathbf{e}_{\text{rel}}^{(L)} \in \mathbb{R}^{d_{\text{rel}}}$, and $\hat{\mathbf{e}} \in \mathbb{R}^{d}$ ($d = d_{\text{att}} + d_{\text{rel}}$).

The embedding concatenation does not merge the attribute- and relation-aware embedding spaces of a KG. For an entity, its two types of representations take up two different parts of its final embedding. In the alignment learning and retrieval, the attribute-aware part of an source entity embedding is computed with the same part of target entity embeddings. The same to the relation-aware embeddings. Therefore, our method can avoid the incoherent representation issue.

4.6 Alignment Learning and Retrieval

The two KGs are encoded jointly in our method. We model entity alignment as a multi-class classification task, where the input is a source entity and the corresponding label is its counterpart entity in the target KG. As entity alignment is often formulated as a one-to-one matching task [30], [35], in our classification problem, each input sample is assigned to one and only one label. Let $\mathcal{E}'$ denote the candidate entity set in the target KG (i.e., classification labels). For source entity e, its alignment probability with $e' \in \mathcal{E}'$ is given by

$$p_{(e,e')} = \frac{\text{sim}(\hat{\mathbf{e}}, \hat{\mathbf{e}}')}{\sum_{e'' \in \mathcal{E}'} \text{sim}(\hat{\mathbf{e}}, \hat{\mathbf{e}}'')}, \tag{16}$$

where $\text{sim}(\hat{\mathbf{e}}, \hat{\mathbf{e}}')$ denotes the embedding similarity between two entities like cosine. Let $\mathbf{q}_e$ be the one-hot label vector for entity e, where only the value at the position of the correct counterpart entity is 1 and others are 0. We adopt the label smoothing strategy as suggested in [40] to get soft labels. For entity e, we use the cross-entropy $\text{CE}()$ to calculate the loss between $\mathbf{q}_e$ and the prediction distribution over all candidates $\mathbf{p}_e$. Formally, the overall loss is defined as

$$\mathcal{L} = \sum_{e \in \mathcal{E}_{\text{train}}} \text{CE}(\mathbf{q}_e, \mathbf{p}_e), \tag{17}$$

where $\mathcal{E}_{\text{train}}$ is the set of the source entities in the training data, i.e., $\mathcal{E}_{\text{train}} = \{e \,|\, (e, e') \in \mathcal{D}_{\text{train}}\}$. $\text{CE}(\mathbf{x}, \mathbf{y}) = -\sum x \log y$ calculates the cross entropy.

Given entity embeddings, entity alignment can be found by the nearest neighbor search. We use a vector similarity metric (e.g., cosine) to calculate the embedding similarities between the source entity and candidate entities. Then, we use the nearest neighbor search to find the most similar candidate as the predicted counterpart for the source entity.

4.7 Discussions on Complexity

We provide an analysis on the space and parameter complexity of our method. In the worst case where the values in all attribute triples have no duplicates, the size of the input (i.e., the value embeddings) is $O(d_{\text{word}} \times |\mathcal{T}_{\text{att}_1}| + d_{\text{word}} \times |\mathcal{T}_{\text{att}_2}|)$. $d_{\text{word}} = 768$ in BERT. The parameter complexity of attribute embedding and entity embedding is $O\big(d_{\text{att}} \times (|\mathcal{A}_1| + |\mathcal{A}_2|) + d_{\text{rel}} \times (|\mathcal{R}_1| + |\mathcal{R}_2|) + d \times (|\mathcal{E}_1| + |\mathcal{E}_2|)\big)$. Considering that $|\mathcal{A}_1| + |\mathcal{A}_2| \approx |\mathcal{R}_1| + |\mathcal{R}_2| \ll |\mathcal{E}_1| + |\mathcal{E}_2| < |\mathcal{T}_{\text{att}_1}| + |\mathcal{T}_{\text{att}_2}|$, the space complexity of our method is generally linear to the number of entities and attribute triples. For alignment retrieval, the nearest neighbor search is the most time-consuming step due to the large candidate space and pair-wise similarity computation. Its complexity is $O(|\mathcal{E}_1| \times |\mathcal{E}_2|)$. For entity alignment in large KGs, we can use the divide-and-conquer algorithms to reduce the complexity [13], or employ advanced implementations such as the efficient similarity search library Faiss [15] for fast nearest neighbor retrieval.

5 Experiments

In this section, we report our experimental results. The source code is publicly available at our GitHub repository.[7]

5.1 Entity Alignment Settings

We consider the following entity alignment settings:

- *Conventional entity alignment* (conventional EA). This is an ideal setting for embedding-based entity alignment. It assumes that both relation and attribute triples are available. Most existing methods confine themselves to this setting. However, it is unable to evaluate the robustness of entity alignment methods. Therefore, we further consider the following challenging settings.
- *Entity alignment without attribute triples* (EA w/o attributes). In this setting, we remove attribute triples and only use relation triples for entity alignment. It is a widely-used setting for relation-based entity alignment methods. We use this setting to evaluate the effectiveness of the proposed relation-aware entity embedding.
- *Entity alignment without relation triples* (EA w/o relations). In this setting, we remove relation triples and only use attribute triples for entity alignment. This setting is motivated by our observation that many entities in real KGs have no relations. To the best of our knowledge, no previous work can handle this challenging setting without manual feature selection or attribute alignment.
- *Entity alignment without relations and names* (EA w/o relations and names). As we have discussed, using entity names brings potential risk of test data leakage and entity names are not always available. We design this setting to evaluate the robustness of entity alignment methods using general attribute triples. It is the most challenging setting that we focus on in this work.

5.2 Benchmark Dataset

We choose the recent benchmark dataset OpenEA [35] in the experiments, for which there are two principal reasons. First, the regular version (V1) of the OpenEA dataset is more approximate to the realistic entity alignment situation than other synthetic datasets, including WK3L [5], DBP15K [31] and DWY100K [32]. The entity degree distribution in the V1 dataset is similar to that in real KGs. By contrast, other datasets contain much more high-degree entities and would give rise to a biased evaluation. Second, the OpenEA dataset contains complete attribute triples that can adequately support the evaluation in our problem setting. By contrast, WK3L and DWY100K do not contain attribute triples, and DBP15K does not provide attribute values. Hence, they are inapplicable to our evaluation. We use the V1 version of each dataset rather than the synthetic version (V2). All these datasets contain both attribute triples and relation triples. In each 15K dataset, 20% of the entity alignment pairs are used as training data, 10% as validation data, and 70% as test data. The split ratio for the 100K datasets is the same as the 15K datasets. We reuse the data split to ensure a fair comparison. Please note that we do not use the D-Y (DBpedia-YAGO) dataset in OpenEA due to its name bias issue as we have discussed in Section 3.2.

7. https://github.com/nju-websoft/RoadEA

TABLE 4: Entity alignment results in the conventional EA setting. **Bold** scores denote the best results.

Methods	EN-FR-15K			EN-DE-15K			D-W-15K			EN-FR-100K			EN-DE-100K			D-W-100K		
	H@1	H@5	MRR	H@1	H@5	MRR	H@1	H@5	MRR	H@1	H@5	MRR	H@1	H@5	MRR	H@1	H@5	MRR
MTransE	.247$_{\pm.006}$	.467$_{\pm.009}$	.351$_{\pm.007}$	.307$_{\pm.007}$	.518$_{\pm.004}$	.407$_{\pm.006}$	.259$_{\pm.008}$	.461$_{\pm.012}$	.354$_{\pm.008}$	.138$_{\pm.002}$	.261$_{\pm.004}$	.202$_{\pm.002}$	.140$_{\pm.003}$	.264$_{\pm.004}$	.204$_{\pm.004}$	.210$_{\pm.003}$	.358$_{\pm.003}$	.282$_{\pm.003}$
IPTransE	.169$_{\pm.013}$	.320$_{\pm.025}$	.243$_{\pm.019}$	.350$_{\pm.009}$	.515$_{\pm.012}$	.430$_{\pm.011}$	.232$_{\pm.012}$	.380$_{\pm.016}$	.303$_{\pm.014}$	.158$_{\pm.004}$	.277$_{\pm.008}$	.219$_{\pm.006}$	.226$_{\pm.014}$	.357$_{\pm.019}$	.292$_{\pm.017}$	.221$_{\pm.004}$	.352$_{\pm.008}$	.285$_{\pm.006}$
AlignE	.357$_{\pm.023}$	.611$_{\pm.025}$	.473$_{\pm.024}$	.552$_{\pm.027}$	.741$_{\pm.020}$	.638$_{\pm.023}$	.406$_{\pm.010}$	.627$_{\pm.009}$	.506$_{\pm.010}$	.294$_{\pm.007}$	.483$_{\pm.008}$	.388$_{\pm.007}$	.423$_{\pm.009}$	.593$_{\pm.009}$	.505$_{\pm.009}$	.385$_{\pm.012}$	.587$_{\pm.014}$	.478$_{\pm.013}$
SEA	.280$_{\pm.015}$	.530$_{\pm.026}$	.397$_{\pm.019}$	.530$_{\pm.027}$	.718$_{\pm.026}$	.617$_{\pm.025}$	.360$_{\pm.012}$	.572$_{\pm.015}$	.458$_{\pm.013}$	.225$_{\pm.011}$	.399$_{\pm.013}$	.314$_{\pm.012}$	.341$_{\pm.016}$	.502$_{\pm.017}$	.421$_{\pm.016}$	.291$_{\pm.012}$	.470$_{\pm.014}$	.378$_{\pm.013}$
RSN4EA	.393$_{\pm.007}$	.595$_{\pm.012}$	.487$_{\pm.009}$	.587$_{\pm.001}$	.752$_{\pm.003}$	.662$_{\pm.001}$	.441$_{\pm.008}$	.615$_{\pm.007}$	.521$_{\pm.007}$	.293$_{\pm.004}$	.452$_{\pm.006}$	.371$_{\pm.004}$	.430$_{\pm.002}$	.570$_{\pm.001}$	.497$_{\pm.001}$	.384$_{\pm.004}$	.533$_{\pm.006}$	.454$_{\pm.005}$
AliNet	.364$_{\pm.005}$	.597$_{\pm.005}$	.467$_{\pm.004}$	.604$_{\pm.007}$	.759$_{\pm.004}$	.673$_{\pm.005}$	.440$_{\pm.007}$	.628$_{\pm.008}$	.522$_{\pm.007}$	.266$_{\pm.003}$	.444$_{\pm.003}$	.348$_{\pm.002}$	.405$_{\pm.002}$	.546$_{\pm.003}$	.471$_{\pm.002}$	.369$_{\pm.002}$	.535$_{\pm.003}$	.444$_{\pm.002}$
KEGCN	.400$_{\pm.009}$	.659$_{\pm.012}$	.515$_{\pm.010}$	.643$_{\pm.005}$	.816$_{\pm.002}$	.719$_{\pm.003}$	.512$_{\pm.004}$	.719$_{\pm.003}$	.603$_{\pm.003}$	.308$_{\pm.002}$	.517$_{\pm.002}$	.408$_{\pm.002}$	.450$_{\pm.001}$	.627$_{\pm.002}$	.534$_{\pm.001}$	.422$_{\pm.001}$	.616$_{\pm.002}$	.511$_{\pm.002}$
JAPE	.262$_{\pm.006}$	.497$_{\pm.010}$	.372$_{\pm.007}$	.288$_{\pm.016}$	.512$_{\pm.018}$	.394$_{\pm.016}$	.250$_{\pm.007}$	.457$_{\pm.010}$	.348$_{\pm.007}$	.165$_{\pm.002}$	.310$_{\pm.002}$	.240$_{\pm.002}$	.152$_{\pm.006}$	.291$_{\pm.009}$	.223$_{\pm.007}$	.211$_{\pm.004}$	.369$_{\pm.004}$	.287$_{\pm.004}$
GCNAlign	.338$_{\pm.002}$	.589$_{\pm.009}$	.451$_{\pm.005}$	.481$_{\pm.003}$	.679$_{\pm.005}$	.571$_{\pm.003}$	.364$_{\pm.009}$	.580$_{\pm.010}$	.461$_{\pm.008}$	.230$_{\pm.002}$	.412$_{\pm.004}$	.319$_{\pm.003}$	.317$_{\pm.007}$	.485$_{\pm.008}$	.399$_{\pm.007}$	.324$_{\pm.002}$	.507$_{\pm.004}$	.409$_{\pm.003}$
IMUSE	.569$_{\pm.006}$	.717$_{\pm.010}$	.638$_{\pm.008}$	.580$_{\pm.017}$	.720$_{\pm.014}$	.647$_{\pm.015}$	.327$_{\pm.016}$	.523$_{\pm.024}$	.419$_{\pm.019}$	.439$_{\pm.002}$	.546$_{\pm.004}$	.492$_{\pm.003}$	.421$_{\pm.005}$	.516$_{\pm.005}$	.469$_{\pm.005}$	.276$_{\pm.010}$	.437$_{\pm.016}$	.355$_{\pm.013}$
AttrE	.481$_{\pm.010}$	.671$_{\pm.009}$	.569$_{\pm.010}$	.517$_{\pm.011}$	.687$_{\pm.013}$	.597$_{\pm.011}$	.299$_{\pm.004}$	.467$_{\pm.003}$	.381$_{\pm.003}$	.403$_{\pm.019}$	.572$_{\pm.019}$	.483$_{\pm.019}$	.399$_{\pm.010}$	.554$_{\pm.012}$	.473$_{\pm.011}$	.209$_{\pm.008}$	.335$_{\pm.011}$	.273$_{\pm.009}$
KDCoE	.581$_{\pm.004}$	.680$_{\pm.004}$	.628$_{\pm.003}$	.529$_{\pm.014}$	.629$_{\pm.015}$	.580$_{\pm.014}$	.247$_{\pm.020}$	.412$_{\pm.029}$	.325$_{\pm.023}$	.482$_{\pm.005}$	.515$_{\pm.006}$	.499$_{\pm.005}$	.506$_{\pm.014}$	.591$_{\pm.019}$	.549$_{\pm.016}$	.157$_{\pm.003}$	.243$_{\pm.007}$	.199$_{\pm.005}$
RDGCN	.755$_{\pm.004}$	**.854**$_{\pm.003}$	.800$_{\pm.003}$	.830$_{\pm.006}$	.895$_{\pm.004}$	.859$_{\pm.005}$	.515$_{\pm.008}$	.669$_{\pm.006}$	.584$_{\pm.007}$	.640$_{\pm.004}$	.732$_{\pm.004}$	.683$_{\pm.004}$	.722$_{\pm.002}$	.794$_{\pm.002}$	.756$_{\pm.002}$	.362$_{\pm.002}$	.485$_{\pm.002}$	.420$_{\pm.002}$
MultiKE	.749$_{\pm.004}$	.819$_{\pm.005}$	.782$_{\pm.004}$	.756$_{\pm.004}$	.809$_{\pm.003}$	.782$_{\pm.003}$	.411$_{\pm.010}$	.521$_{\pm.017}$	.468$_{\pm.012}$	.629$_{\pm.002}$	.680$_{\pm.002}$	.655$_{\pm.002}$	.668$_{\pm.002}$	.712$_{\pm.002}$	.690$_{\pm.001}$	.290$_{\pm.006}$	.357$_{\pm.011}$	.326$_{\pm.009}$
RoadEA	**.810**$_{\pm.006}$	**.854**$_{\pm.004}$	**.831**$_{\pm.005}$	**.868**$_{\pm.009}$	**.905**$_{\pm.008}$	**.886**$_{\pm.010}$	**.653**$_{\pm.003}$	**.789**$_{\pm.002}$	**.714**$_{\pm.002}$	**.743**$_{\pm.002}$	**.799**$_{\pm.004}$	**.769**$_{\pm.003}$	**.831**$_{\pm.001}$	**.868**$_{\pm.005}$	**.847**$_{\pm.006}$	**.572**$_{\pm.007}$	**.694**$_{\pm.008}$	**.629**$_{\pm.007}$

5.3 Baseline Methods

We compare RoadEA with fourteen popular entity alignment methods from previous work. They include (i) relation-based methods MTransE [5], IPTransE [58], AlignE [32], SEA [26], RSN4EA [11], AliNet [34] and KEGCN [51]; (ii) attribute-enhanced methods JAPE [31], GCNAlign [42], IMUSE [12], AttrE [37] and KDCoE [4]; and (iii) name-enhanced methods RDGCN [44] and MultiKE [55]. Please refer to Section 2 for more details about these baseline methods. For a fair comparison, we reuse the OpenEA's implementations of these baselines expect KEGCN. We reuse the official code of KEGCN to produce its results on the OpenEA dataset with careful hyper-parameter tuning. We do not compare with JarKA [3] and EMGCN [25] because they use the alignment signals from machine translation while our method does not. We also do not consider some modern semi-supervised methods such as BootEA [32] and MRAEA [23] that augment the seed entity alignment iteratively, because the baselines and our method are all in the supervised setting.

5.4 Implementation Details

We initialize trainable parameters with the Xavier initializer [10], and optimize the loss in Eq. (17) using Adam [17]. We search among a range for hyper-parameter values, e.g., the learning rate in $\{0.00005, 0.0001, 0.0005, 0.001\}$, the layer number in $\{1, 2, 3, 4\}$. We use early stop to terminate training based on the MRR score tested every 20 epochs on the validation data. The selected settings for hyper-parameters are reported as follows. The embedding dimensions are $d_{\text{word}} = 768$, $d_{\text{val}} = d_{\text{att}} = 32$, $d_{\text{rel}} = 320$, and $d = 352$. The learning rate is 0.0001. We use two layers (i.e., $L = 2$) for embedding aggregation. The number of convolution kernels $c = 2$. The batch size is 1024 for the 15K dataset, and $4,096$ for the 100K dataset. For each entity, the maximum number of attribute triples in attribute aggregation is 15, and the maximum number of relation triples for relational neighbor aggregation is 20. If an entity has more than 15 attribute triples (or 20 relation triples), we randomly sample attribute (or relation) triples for training in each epoch. We use Cosine to measure embedding similarities. We use bert-as-service (base model, uncased) [47] to generate value embeddings.

Following the convention, the default alignment direction is from left to right. Taking D-W as an example, we regard DBpedia as the source KG and seek to find the counterparts of source entities in the target KG Wikidata. As introduced in Section 2.1, we report the H@1, H@5 and MRR scores to assess entity alignment performance. Following OpenEA, we report the average results of five-fold cross-validation. For baseline methods, in the conventional EA setting, we directly reuse the reported results in the OpenEA paper. In other settings, we try our best to tune them to obtain good results.

5.5 Results and Analyses

We first present the results under different settings in Sect. 5.5.1. Then in Sect 5.5.2, we analyze the results from six perspectives to gain a clear and thorough understanding. Finally, we summarize the experimental findings in Sect. 5.5.3.

5.5.1 Results in Different Settings

Conventional EA. In this setting, we compare our method RoadEA against all baselines. Table 4 presents the results. We observe that RoadEA achieves significantly improved performance. It outperforms all three groups of baselines by a large margin in terms of all metrics and datasets. For example, RoadEA improves the second-best scores by 17.6% on H@1, 7.0% on H@5, and 12.2% on MRR, averagely. This is because RoadEA can fully leverage relation-aware and attribute-aware embeddings for alignment learning, and let them benefit each other.

EA w/o attributes. In this setting, the initial entity representations for relation-aware embedding are randomly generated rather than learned from attribute triples. We compare this variant with the seven relation-based entity alignment methods: MTransE, IPTransE, BootEA, SEA, RSN4EA, AliNet and KEGCN. We also choose the attribute-enhanced methods JAPE, GCNAlign, IMUSE, KDCoE, AttrE, RDGCN and MultiKE as baselines. We remove all attribute triples from the datasets to run these baselines and get their results. Table 5 shows the results. RoadEA still outperforms six relation-based methods (expect KEGCN on D-W-15K) and all attribute-enhanced baselines on all datasets. The average performance improvement compared to the second-best scores are 21.1%, 10.9% and 14.8% on H@1, H@5, and MRR, respectively. This demonstrates the effectiveness of our relation-aware entity embedding module. Comparing the results in Tables 4 and 5, we further find that RoadEA increases the lead from attribute-enhanced methods when only using relation triples. For example, on EN-FR-15K of

TABLE 5: Entity alignment results in the setting of EA w/o attributes.

Methods	EN-FR-15K			EN-DE-15K			D-W-15K			EN-FR-100K			EN-DE-100K			D-W-100K		
	H@1	H@5	MRR	H@1	H@5	MRR	H@1	H@5	MRR	H@1	H@5	MRR	H@1	H@5	MRR	H@1	H@5	MRR
JAPE	.251$_{\pm.001}$	.479$_{\pm.005}$	.358$_{\pm.002}$	.302$_{\pm.011}$	.527$_{\pm.013}$	.408$_{\pm.011}$	.255$_{\pm.001}$	.460$_{\pm.002}$	.352$_{\pm.001}$	.164$_{\pm.001}$	.308$_{\pm.001}$	.238$_{\pm.001}$	.153$_{\pm.002}$	.292$_{\pm.002}$	.224$_{\pm.002}$	.211$_{\pm.005}$	.368$_{\pm.007}$	.287$_{\pm.006}$
GCNAlign	.323$_{\pm.001}$	.568$_{\pm.001}$	.434$_{\pm.002}$	.470$_{\pm.006}$	.664$_{\pm.004}$	.558$_{\pm.005}$	.350$_{\pm.001}$	.564$_{\pm.009}$	.445$_{\pm.004}$	.213$_{\pm.006}$	.385$_{\pm.008}$	.298$_{\pm.006}$	.298$_{\pm.008}$	.460$_{\pm.007}$	.377$_{\pm.008}$	.299$_{\pm.003}$	.477$_{\pm.007}$	.382$_{\pm.005}$
IMUSE	.256$_{\pm.002}$	.471$_{\pm.007}$	.358$_{\pm.004}$	.473$_{\pm.007}$	.646$_{\pm.006}$	.555$_{\pm.006}$	.321$_{\pm.005}$	.509$_{\pm.006}$	.410$_{\pm.005}$	.212$_{\pm.003}$	.367$_{\pm.005}$	.292$_{\pm.004}$	.333$_{\pm.005}$	.477$_{\pm.006}$	.406$_{\pm.005}$	.286$_{\pm.001}$	.444$_{\pm.001}$	.363$_{\pm.001}$
KDCoE	.186$_{\pm.001}$	.337$_{\pm.002}$	.260$_{\pm.002}$	.345$_{\pm.008}$	.505$_{\pm.004}$	.422$_{\pm.006}$	.220$_{\pm.010}$	.355$_{\pm.011}$	.285$_{\pm.013}$	.160$_{\pm.004}$	.271$_{\pm.008}$	.217$_{\pm.006}$	.248$_{\pm.004}$	.372$_{\pm.005}$	.311$_{\pm.005}$	.227$_{\pm.007}$	.349$_{\pm.011}$	.286$_{\pm.009}$
AttrE	.234$_{\pm.002}$	.438$_{\pm.001}$	.332$_{\pm.002}$	.511$_{\pm.006}$	.680$_{\pm.008}$	.590$_{\pm.006}$	.322$_{\pm.009}$	.510$_{\pm.008}$	.410$_{\pm.010}$	.213$_{\pm.006}$	.367$_{\pm.008}$	.292$_{\pm.007}$	.337$_{\pm.010}$	.483$_{\pm.011}$	.410$_{\pm.011}$	.287$_{\pm.005}$	.446$_{\pm.006}$	.365$_{\pm.005}$
RDGCN	.255$_{\pm.004}$	.476$_{\pm.009}$	.355$_{\pm.003}$	.511$_{\pm.004}$	.694$_{\pm.003}$	.592$_{\pm.002}$	.331$_{\pm.001}$	.510$_{\pm.008}$	.409$_{\pm.003}$	.158$_{\pm.006}$	.264$_{\pm.008}$	.209$_{\pm.005}$	.309$_{\pm.003}$	.413$_{\pm.005}$	.358$_{\pm.002}$	.273$_{\pm.005}$	.381$_{\pm.004}$	.322$_{\pm.004}$
MultiKE	.337$_{\pm.005}$	.580$_{\pm.010}$	.449$_{\pm.008}$	.576$_{\pm.003}$	.770$_{\pm.003}$	.662$_{\pm.001}$	.403$_{\pm.010}$	.639$_{\pm.013}$	.506$_{\pm.011}$	.269$_{\pm.004}$	.454$_{\pm.004}$	.361$_{\pm.003}$	.378$_{\pm.004}$	.550$_{\pm.002}$	.462$_{\pm.003}$	.356$_{\pm.003}$	.557$_{\pm.005}$	.450$_{\pm.004}$
RoadEA	**.447**$_{\pm.006}$	**.683**$_{\pm.002}$	**.552**$_{\pm.004}$	**.669**$_{\pm.004}$	**.834**$_{\pm.004}$	**.742**$_{\pm.004}$	**.495**$_{\pm.005}$	**.694**$_{\pm.003}$	**.584**$_{\pm.003}$	**.303**$_{\pm.001}$	**.492**$_{\pm.001}$	**.393**$_{\pm.001}$	**.458**$_{\pm.008}$	**.620**$_{\pm.007}$	**.533**$_{\pm.007}$	**.432**$_{\pm.004}$	**.611**$_{\pm.003}$	**.513**$_{\pm.003}$

TABLE 6: Entity alignment results in the setting of EA w/o relations.

Methods	EN-FR-15K			EN-DE-15K			D-W-15K			EN-FR-100K			EN-DE-100K			D-W-100K		
	H@1	H@5	MRR	H@1	H@5	MRR	H@1	H@5	MRR	H@1	H@5	MRR	H@1	H@5	MRR	H@1	H@5	MRR
AttrE	.151$_{\pm.006}$	.286$_{\pm.008}$	.219$_{\pm.007}$	.155$_{\pm.005}$	.273$_{\pm.001}$	.214$_{\pm.002}$	.036$_{\pm.001}$	.092$_{\pm.003}$	.068$_{\pm.001}$	.084$_{\pm.004}$	.171$_{\pm.004}$	.129$_{\pm.004}$	.113$_{\pm.001}$	.194$_{\pm.002}$	.153$_{\pm.001}$	.012$_{\pm.002}$	.035$_{\pm.004}$	.026$_{\pm.002}$
MultiKE	.704$_{\pm.002}$	.728$_{\pm.004}$	.716$_{\pm.003}$	.713$_{\pm.003}$	.739$_{\pm.005}$	.726$_{\pm.004}$	.324$_{\pm.004}$	.335$_{\pm.005}$	.331$_{\pm.003}$	.628$_{\pm.004}$	.645$_{\pm.002}$	.636$_{\pm.003}$	.670$_{\pm.004}$	.685$_{\pm.002}$	.677$_{\pm.004}$	.223$_{\pm.001}$	.228$_{\pm.001}$	.226$_{\pm.001}$
RoadEA	**.747**$_{\pm.001}$	**.787**$_{\pm.003}$	**.766**$_{\pm.001}$	**.808**$_{\pm.001}$	**.850**$_{\pm.003}$	**.828**$_{\pm.002}$	**.376**$_{\pm.003}$	**.423**$_{\pm.003}$	**.403**$_{\pm.000}$	**.695**$_{\pm.004}$	**.751**$_{\pm.004}$	**.722**$_{\pm.004}$	**.745**$_{\pm.005}$	**.795**$_{\pm.004}$	**.769**$_{\pm.005}$	**.289**$_{\pm.003}$	**.349**$_{\pm.002}$	**.319**$_{\pm.003}$

the conventional EA setting, RoadEA outperforms MultiKE on H@1 by 8.14%, but the improvement extends to 32.6% in the setting of EA w/o attributes. This shows the strength of our method in making use of relation triples.

EA w/o relations. We compare RoadEA against AttrE and MultiKE because they are the only two baseline methods that can work without relation triples. Table 6 presents the results. RoadEA outperforms AttrE and MultiKE in this setting. Moreover, the performance of AttrE and MultiKE declines radically on a few datasets compared with that in the conventional EA setting. For example, the H@1 results of AttrE are less than 0.16 on all datasets. By contrast, the performance of RoadEA does not decline too much. This is because removing relation triples reduces RoadEA's performance in aligning entities with rich relations, but the number of these entities only accounts for a small proportion. MultiKE also does not suffer severe performance degradation in this setting, because it uses entity names as an important feature view, and the strong alignment signals in names guarantees its performance. However, as previously stated, such performance is not robust. RoadEA has no special treatment of entity names, but still outperforms MultiKE. The results indicate that RoadEA is robust against the unavailability of relation triples. The performance drop on D-W-15K is dramatic compared to that on other two datasets. This is because the attribute triples in D-W are highly heterogeneous, significantly increasing the difficulty of aligning entities solely based on attribute triples.

EA w/o relations and names. This is more challenging than the above setting of EA w/o relations. The only available information is the general attribute triples. To the best of our knowledge, no previous work has considered this setting. For a clearer comparison, we show in Fig. 6 the H@1 results and the performance reduction compared against those under the conventional EA setting as listed in Table 4. Both AttrE and MultiKE fail to work under this setting. Specifically, the H@1 scores of AttrE and MultiKE are both less than 0.02, which decline radically with the drop ratio of more than 94.0% on all datasets compared with the conventional setting. Compared against the setting w/o relations, the drop ratio reaches more than 90.0%. These results indicate that AttrE and MultiKE cannot take advantage of the general attribute triples for entity alignment, and they highly depend on name triples. Although the H@1 results of our method RoadEA also suffer from a decline, they are still much better than AttrE and MultiKE. On the most heterogeneous dataset D-W, RoadEA still shows promising performance. This experiment demonstrates the excellent robustness of RoadEA.

5.5.2 Further Analyses

Impact of dataset heterogeneity. Name-enhanced methods perform worse on D-W than on EN-FR and EN-DE. This is because the D-W datasets contain a much smaller proportion of name attributes but have greater attribute heterogeneity than the other two datasets. It is a big challenge for name-enhanced methods like RDGCN and MultiKE. By contrast, RoadEA still shows promising performance on D-W. Our improvement over name-enhanced methods is more significant on D-W than on the other datasets. Our method does not rely on names, and can effectively leverage other general attribute triples. Besides, on EN-FR and EN-DE, attribute-enhanced methods generally outperform relation-based ones but fall behind name-enhanced methods. It is nearly the opposite on the D-W datasets. This finding also indicates the importance of attributes including names for entity alignment.

Impact of dataset scale. The performance of our method and also other baselines on 100K datasets is lower than that on the corresponding 15K datasets. This is because a 100K dataset has a much larger candidate space than a 15K dataset. It is difficult to rank the correct counterpart at the top place from an extensive candidate space.

Impact of relation and attribute triples. To further study the impact of the absence of attributes or relations, we compare RoadEA with AttrE and MultiKE in the settings where $K\%$ relation or attribute triples removed. Table 7 displays the results with $K = 0, 50, 100$ on the EN-FR datasets. We do not compare with name-enhanced methods as they are not applicable in this setting. In general, removing more attribute or relation triples causes all three methods to perform worse. MultiKE relies heavily on attribute triples, including name triples. Removing some attribute triples can significantly reduce its performance, whereas removing relation triples has little effect. Although AttrE can leverage both attribute and relation triples, its performance is not promising when some attributes or relations are removed. RoadEA outperforms AttrE and MultiKE in all settings, showing its robustness to

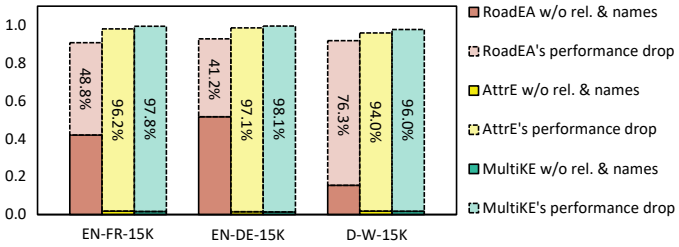


Fig. 6: Average H@1 of AttrE, MultiKE and RoadEA w/o relations or names, along with their performance reduction ratio compared with that in the conventional setting.

TABLE 7: Average H@1 results when $K\%$ triples removed.

Dataset	EN-FR-15K					
Removed trip.	None	100% attr.	50% attr.	50% attr. & rel.	50% rel.	100% rel
AttrE	.481	.234	.259	.149	.322	.151
MultiKE	.749	.337	.433	.416	.681	.704
RoadEA	**.810**	**.447**	**.460**	**.427**	**.775**	**.747**
Dataset	EN-FR-100K					
AttrE	.403	.213	.240	.118	.275	.084
MultiKE	.629	.269	.302	.273	.622	.628
RoadEA	**.743**	**.303**	**.331**	**.312**	**.704**	**.695**

the absence of attributes or relations. In general, attribute triples contribute more than relation triples in RoadEA.

Robustness to the unavailability of names. We compare RoadEA with AttrE, RDGCN and MultiKE in the setting without entity names. Table 8 presents the results. We remove name triples from the datasets for AttrE and our method, leaving only general attribute triples and relation triples. We disable the name view in MultiKE and replace the name-based initialization in RDGCN with randomly generated representations. The performance of AttrE, RDGCN, and MultiKE drops a lot compared to their results in the conventional setting. For instance, the H@1 scores of AttrE, RDGCN and MultiKE on 15K datasets drop by 38.3%, 46.8% and 40.0% on average, respectively. On 100K datasets, their performance declines by 52.2%, 52.4% and 41.7% on average, respectively. Interestingly, their performance is even poorer than many relation-based methods. This shows that many attribute-enhanced methods rely excessively on the name information, which is defectively biased. Inevitably, RoadEA also suffers from a performance decline, with a reduction ratio of 27.8% on H@1 averagely. However, it still outperforms attribute-enhanced methods.

Effectiveness in long-tail entity alignment. To investigate the effectiveness of our method in handling long-tail entity alignment, we divide the entity alignment pairs in the test set into multiple groups based on the number of their involved relation or attribute triples. Fig. 7 compares the H@1 scores of RoadEA and its two variants in each group. We can see from Fig. 7 (a) that most of the entity alignment pairs involve a few relation triples. For example, there are 70.6% of entity alignment pairs, in which each entity has no more than five relation triples. RoadEA w/o attributes performs poorly in aligning the long-tail entities. Through the use of attribute triples, both RoadEA and RoadEA w/o relations can effectively handle long-tail entity alignment. This is where our improvement comes. The H@1 score of RoadEA in aligning entities with degrees within the interval $(0, 2]$ reaches 0.869 on EN-FR-15K, much higher than the

TABLE 8: Entity alignment results without name triples.

Methods	EN-FR-15K			EN-DE-15K			D-W-15K		
	H@1	H@5	MRR	H@1	H@5	MRR	H@1	H@5	MRR
AttrE	.234	.422	.327	.310	.499	.404	.229	.397	.312
RDGCN	.255	.476	.355	.511	.694	.592	.331	.510	.409
MultiKE	.349	.585	.458	.421	.638	.522	.319	.519	.414
RoadEA	**.525**	**.640**	**.579**	**.709**	**.798**	**.750**	**.458**	**.630**	**.538**
	EN-FR-100K			EN-DE-100K			D-W-100K		
AttrE	.140	.252	.198	.142	.256	.200	.153	.261	.208
RDGCN	.158	.264	.209	.309	.413	.358	.273	.381	.322
MultiKE	.279	**.465**	.372	.340	.507	.419	.231	.426	.330
RoadEA	**.369**	.464	**.412**	**.573**	**.668**	**.618**	**.310**	**.434**	**.370**

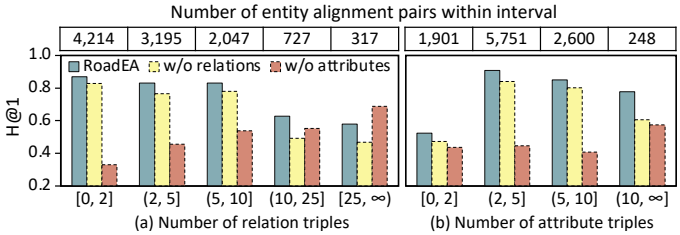


Fig. 7: H@1 in different test alignment groups on EN-FR-15K.

baselines in Fig. 2. As shown in Fig. 7 (b), the long-tail issue also exists in attribute triples where most entity alignment pairs involve a few attribute triples. Our method and variants produce promising results when aligning entities with no more than two attributes. This experiment demonstrates the effectiveness of our method in dealing with the long-tail issue and the complementarity of relations and attributes.

Effectiveness in combining relations and attributes. From the results in the four settings, we also notice that MultiKE achieves comparable and even better performance without using attribute triples on the D-W datasets. This is because the attributes in D-W are too heterogeneous for embedding learning and have a negative impact on directly combining attribute-based and relation-based embeddings in MultiKE. By contrast, RoadEA achieves better performance than its variants without relations or attributes on all datasets. This shows that our relation-aware and attribute-aware encoders can benefit from each other to get improved performance.

5.5.3 *Summary*

The good performance of RoadEA is due to its ability to fully utilize relation and attribute triples, as well as its robustness to the unavailability of entity names. When entity names are available, attribute-aware embeddings contribute more than relation-aware embeddings. When entity names are unavailable, relation-aware embeddings contribute more if the attribute heterogeneity is great (as in D-W). Otherwise, attribute-aware embeddings play a more important role. RoadEA performs well in aligning long-tail entities via the adaptive fusion of relation- and attribute-aware embeddings.

5.6 Ablation Studies

We evaluate the effectiveness of RoadEA's technical parts.

5.6.1 *Effectiveness of the Gating Mechanism*

RoadEA employs a gating mechanism to aggregate the neighbor representations selectively. To verify its effectiveness, we compare it with a degraded variant without the gating mechanism (i.e., RoadEA w/o gating). The variant directly

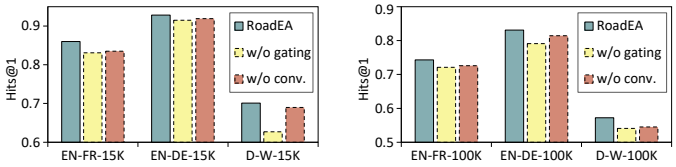


Fig. 8: Performance comparison of RoadEA w/ and w/o the gating mechanism and convolution operator.

aggregates all the neighbor representations without preference and combines them with the central entity embedding as the output. We can see from Fig. 8 that the performance of the degraded variant decreases in varying degree compared with the full version of RoadEA. Moreover, the performance reduction on the D-W datasets is more significant than that on EN-FR and EN-DE. This is because the entity heterogeneity in the EN-DE and EN-FR datasets is small, and there is no need for too much feature selection when aggregating the neighbor entity embeddings. While in D-W datasets, aligned entities between DBpedia and Wikidata have very different attributes, making the direct combination of neighboring entity representations bring substantial adverse effects on alignment learning. This experiment shows the effectiveness and robustness of the gating mechanism used by RoadEA.

5.6.2 Effectiveness of the Convolution Operation

RoadEA uses value convolution to represent an attribute. To evaluate its effectiveness, we compare it with an alternative method that first uses a fully-connected layer for value embedding transformation and then averages these embeddings to represent an attribute. This is equivalent to a degraded variant of our method without the Conv() function in Eq. (5), and is denoted as RoadEA w/o conv. The results are also shown in Fig. 8. We can see that removing the convolution operation results in a decrease in performance on all datasets. For example, on EN-FR-15K, the H@1 score declines from 0.810 to 0.785 when removing value convolution. The results demonstrate the effectiveness value convolution. The reason is that convolution is a good feature extractor to learn high-level representations from value embeddings.

5.6.3 Impact of Value Embeddings

We use BERT [8] to encode attribute values due to its outstanding performance in various tasks. To explore whether RoadEA can still achieve advanced performance using other word embeddings, we replace BERT with another popular word encoder fastText [16]. We consider the setting of EA w/o relations, which is suitable for evaluating the impact of value embeddings. Table 9 presents the results. Using fastText indeed causes reduced performance of RoadEA. For example, on 15K datasets, the average performance drops on H@1, H@5 and MRR are 10.2%, 11.2%, and 10.9%, respectively. The results are broadly in line with expectations as fastText is weaker than BERT in representing literal words. However, RoadEA without BERT still outperforms AttrE and MultiKE in Table 6. This experiment demonstrates the generalization of RoadEA, which does not entirely rely on BERT.

5.6.4 Summary

Our method relies on the attention mechanism to aggregate attribute or relation embeddings. The gating mechanism and value convolution are both effective and can improve H@1 by 0.01 to 0.07. The BERT encoder for value embeddings plays a big role. It improves H@1 by 0.04 to 0.11 when compared to fastText, but RoadEA still works well without BERT.

TABLE 9: Entity alignment results using different value embeddings in the w/o relation setting.

Methods	EN-FR-15K			EN-DE-15K			D-W-15K		
	H@1	H@5	MRR	H@1	H@5	MRR	H@1	H@5	MRR
RoadEA (fastText)	.673	.710	.692	.739	.789	.762	.341	.368	.356
RoadEA (BERT)	**.747**	**.787**	**.766**	**.808**	**.850**	**.828**	**.376**	**.423**	**.403**
	EN-FR-100K			EN-DE-100K			D-W-100K		
RoadEA (fastText)	.617	.699	.656	.640	.706	.671	.193	.229	.211
RoadEA (BERT)	**.695**	**.751**	**.722**	**.745**	**.795**	**.769**	**.289**	**.349**	**.319**

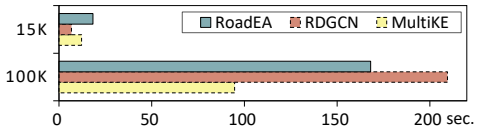


Fig. 9: Average training time of one epoch on EN-FR datasets.

5.7 Running Time Comparison

We compare the running time of RoadEA with RDGCN and MultiKE in Fig. 9. We show the average training time of one epoch on EN-FR datasets, and observe similar results on other datasets. The experiment is conducted on a workstation with an Intel Xeon E3 3.3GHz CPU and 256GB memory. We follow RDGCN and do not use GPUs due to the limited graphics memory that does not have sufficient capacity for word embeddings. On EN-FR-15K, RoadEA costs more time (18.3s) for one-epoch training than MultiKE (12.2s) and RDGCN (6.7s). The reason lies in that our method is deeper than MultiKE and uses more features than RDGCN. MultiKE is a multi-view method, and two views (three in total) use shallow models for embedding learning, leading to high efficiency. RDGCN does not introduce attribute triples for embedding learning. On the 100K dataset, RDGCN becomes the slowest method, and MultiKE is still the fastest one. The most time-consuming part of RDGCN is the kNN-based negative sampling. Its time complexity is $O(|\mathcal{E}_1| \times |\mathcal{E}_2|)$, and would cost much time during training large KGs. RoadEA does not use negative sampling, and shows comparable performance in running time, although it employs comprehensive features. In general, our method gets a good trade-off between effectiveness and efficiency.

5.8 Case Study

To further investigate how our method can align entities based on attribute information, we give a case study on two aligned entities referring to "Star Wars: The Clone Wars" in DBpedia[8] and Wikidata[9]. The two aligned entities are not correctly found when only using relation triples for model training. We find that our attribute triple embeddings can automatically capture some helpful attribute alignment and value alignment to identify the two entities. Fig. 10 shows a visualization of the attribute-value alignment automatically found by the greedy similarity search. The colored block

8. https://dbpedia.org/page/Clone_Wars_(Star_Wars)
9. https://www.wikidata.org/wiki/Q616313

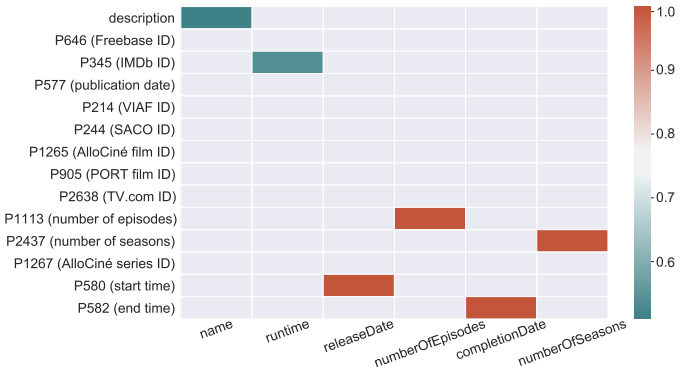


Fig. 10: Value embedding similarity of the automatically found attribute alignment of entity "Star Wars: The Clone Wars" in Wikidata and DBpedia. The Y-axis gives attributes in Wikidata while the X-axis denotes attributes in DBpedia. Gray blocks mean no attribute alignment.

indicates that the corresponding attributes (along with their values) are found similar (green blocks) and even identical (red blocks). In Wikidata, each attribute is encoded by an ID starting with "P". Such surface names (a.k.a., localnames) of attributes are meaningless and cannot be used to find attribute alignment as in some attribute-enhanced methods such as IMUSE [12] and EMGCN [25]. For easy reading, we label attributes with their human-readable names in brackets. Although we do not train RoadEA with any attribute alignment, the method can still learn similar representations for semantically similar attributes, such as P580 (*start time*) in Wikidata and *releaseDate* in DBpedia, as well as P582 (*end time*) in Wikidata and *completionDate* in DBpedia. This is a significant difference between RoadEA and other methods that leverage attributes in an offline way and rely on the symbolic representations of attributes and values to calculate similarities. The significant attribute heterogeneity results in all models' poor performance on D-W. Moreover, although the name information is not found in Wikidata, its description is fortunately aligned to the name in DBpedia. The similarity of their corresponding value embeddings is relatively low, but such alignment also contributes to the entity alignment learning. Besides, the attribute values of other aligned attributes have high similarities. They together help the method learn similar embeddings for the two entities.

6 Conclusions

In this paper, we revisit embedding-based entity alignment methods by conducting extensive experiments to study the impact of entity names, attribute and relation triples on entity alignment. Our findings reveal that relation-based methods cannot effectively handle long-tail entities, and attribute-enhanced methods rely heavily on the name information. We further present RoadEA, a robust and adaptive entity alignment method. It employs attribute and relation encoders to learn entity embeddings by attention mechanisms. The two encoders are combined using a gating mechanism for adaptive embedding fusion. RoadEA does not require the availability of attribute or relation triples. Our experiments in four entity alignment settings demonstrate the effectiveness and robustness of our method. For future work, we plan to extend our method to other alignment tasks, such as ontology matching and entity resolution in databases.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 61872172), and the Collaborative Innovation Center of Novel Software Technology & Industrialization. Zequn Sun is grateful for the support of Program A for Outstanding PhD Candidates of Nanjing University.

References

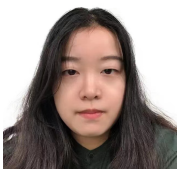
[1] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *NeurIPS*, 2013.
[2] Y. Cao, Z. Liu, C. Li, Z. Liu, J. Li, and T. Chua. Multi-channel graph neural network for entity alignment. In *ACL*, 2019.
[3] B. Chen, J. Zhang, X. Tang, H. Chen, and C. Li. Jarka: Modeling attribute interactions for cross-lingual knowledge alignment. In *PAKDD*, 2020.
[4] M. Chen, Y. Tian, K. Chang, S. Skiena, and C. Zaniolo. Co-training embeddings of knowledge graphs and entity descriptions for cross-lingual entity alignment. In *IJCAI*, 2018.
[5] M. Chen, Y. Tian, M. Yang, and C. Zaniolo. Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. In *IJCAI*, 2017.
[6] M. Chen, T. Zhou, P. Zhou, and C. Zaniolo. Multi-graph affinity embeddings for multilingual knowledge graphs. In *AKBC*, 2017.
[7] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel. Convolutional 2d knowledge graph embeddings. In *AAAI*, 2018.
[8] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.
[9] C. Ge, X. Liu, L. Chen, B. Zheng, and Y. Gao. Make it easy: An effective end-to-end entity alignment framework. In *SIGIR*, 2021.
[10] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010.
[11] L. Guo, Z. Sun, and W. Hu. Learning to exploit long-term relational dependencies in knowledge graphs. In *ICML*, 2019.
[12] F. He, Z. Li, Q. Yang, A. Liu, G. Liu, P. Zhao, L. Zhao, M. Zhang, and Z. Chen. Unsupervised entity alignment using attribute triples and relation triples. In *DASFAA*, 2019.
[13] W. Hu, Y. Qu, and G. Cheng. Matching large ontologies: A divide-and-conquer approach. *Data & Knowledge Engineering*, 67(1), 2008.
[14] X. Huang, J. Zhang, D. Li, and P. Li. Knowledge graph embedding based question answering. In *WSDM*, 2019.
[15] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3), 2021.
[16] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. Bag of tricks for efficient text classification. In *EACL*, 2017.
[17] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.
[18] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.
[19] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer. DBpedia – a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, 6(2), 2015.
[20] F. Liu, M. Chen, D. Roth, and N. Collier. Visual pivoting for (unsupervised) entity alignment. *CoRR*, abs/2009.13603, 2020.
[21] Z. Liu, Y. Cao, L. Pan, J. Li, and T. Chua. Exploring and evaluating attributes, values, and structures for entity alignment. In *EMNLP*, 2020.
[22] A. L. Maas, A. Y. Hannun, A. Y. Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In *ICML*, 2013.
[23] X. Mao, W. Wang, H. Xu, M. Lan, and Y. Wu. MRAEA: an efficient and robust entity alignment approach for cross-lingual knowledge graph. In *WSDM*, 2020.
[24] X. Mao, W. Wang, H. Xu, Y. Wu, and M. Lan. Relational reflection entity alignment. In *CIKM*, 2020.
[25] T. T. Nguyen, T. T. Huynh, H. Yin, V. V. Tong, D. Sakong, B. Zheng, and Q. V. H. Nguyen. Entity alignment for knowledge graphs with multi-order convolutional networks. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[26] S. Pei, L. Yu, R. Hoehndorf, and X. Zhang. Semi-supervised entity alignment via knowledge graph embedding with awareness of degree difference. In *WWW*, 2019.
[27] T. Rebele, F. M. Suchanek, J. Hoffart, J. Biega, E. Kuzey, and G. Weikum. YAGO: A multilingual knowledge base from wikipedia, wordnet, and geonames. In *ISWC*, 2016.
[28] M. S. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling. Modeling relational data with graph convolutional networks. In *ESWC*, 2018.
[29] R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. *CoRR*, 2015.
[30] F. M. Suchanek, S. Abiteboul, and P. Senellart. PARIS: probabilistic alignment of relations, instances, and schema. *PVLDB*, 5(3), 2011.
[31] Z. Sun, W. Hu, and C. Li. Cross-lingual entity alignment via joint attribute-preserving embedding. In *ISWC*, 2017.
[32] Z. Sun, W. Hu, Q. Zhang, and Y. Qu. Bootstrapping entity alignment with knowledge graph embedding. In *IJCAI*, 2018.
[33] Z. Sun, J. Huang, W. Hu, M. Chen, L. Guo, and Y. Qu. Transedge: Translating relation-contextualized embeddings for knowledge graphs. In *ISWC*, 2019.
[34] Z. Sun, C. Wang, W. Hu, M. Chen, J. Dai, W. Zhang, and Y. Qu. Knowledge graph alignment network with gated multi-hop neighborhood aggregation. In *AAAI*, 2020.
[35] Z. Sun, Q. Zhang, W. Hu, C. Wang, M. Chen, F. Akrami, and C. Li. A benchmarking study of embedding-based entity alignment for knowledge graphs. *PVLDB*, 2020.
[36] X. Tang, J. Zhang, B. Chen, Y. Yang, H. Chen, and C. Li. BERT-INT: A bert-based interaction model for knowledge graph alignment. In *IJCAI*, 2020.
[37] B. D. Trisedya, J. Qi, and R. Zhang. Entity alignment between knowledge graphs using attribute embeddings. In *AAAI*, 2019.
[38] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *ICLR*, 2018.
[39] D. Vrandecic and M. Krotzsch. Wikidata: A free collaborative knowledgebase. *Communications of the ACM*, 57(10), 2014.
[40] Q. Wang, P. Huang, H. Wang, S. Dai, W. Jiang, J. Liu, Y. Lyu, Y. Zhu, and H. Wu. Coke: Contextualized knowledge graph embedding. *CoRR*, abs/1911.02168, 2019.
[41] X. Wang, X. He, Y. Cao, M. Liu, and T. Chua. KGAT: knowledge graph attention network for recommendation. In *KDD*, 2019.
[42] Z. Wang, Q. Lv, X. Lan, and Y. Zhang. Cross-lingual knowledge graph alignment via graph convolutional networks. In *EMNLP*, 2018.
[43] A. Weichselbraun, P. Kuntschik, and A. M. P. Brasoveanu. Name variants for improving entity discovery and linking. In *LDK*, 2019.
[44] Y. Wu, X. Liu, Y. Feng, Z. Wang, R. Yan, and D. Zhao. Relation-aware entity alignment for heterogeneous knowledge graphs. In *IJCAI*, 2019.
[45] Y. Wu, X. Liu, Y. Feng, Z. Wang, and D. Zhao. Jointly learning entity and relation representations for entity alignment. In *EMNLP*, 2019.
[46] Y. Wu, X. Liu, Y. Feng, Z. Wang, and D. Zhao. Neighborhood matching network for entity alignment. In *ACL*, 2020.
[47] H. Xiao. bert-as-service. https://github.com/hanxiao/bert-as-service, 2018.
[48] C. Xiong, R. Power, and J. Callan. Explicit semantic ranking for academic search via knowledge graph embedding. In *WWW*, 2017.
[49] K. Xu, L. Wang, M. Yu, Y. Feng, Y. Song, Z. Wang, and D. Yu. Cross-lingual knowledge graph alignment via graph matching neural network. In *ACL*, 2019.
[50] H. Yang, Y. Zou, P. Shi, W. Lu, J. Lin, and X. Sun. Aligning cross-lingual entities with multi-aspect information. In *EMNLP*, 2019.
[51] D. Yu, Y. Yang, R. Zhang, and Y. Wu. Knowledge embedding based graph convolutional network. In *WWW*, 2021.
[52] W. Zeng, X. Zhao, J. Tang, X. Li, M. Luo, and Q. Zheng. Towards entity alignment in the open world: An unsupervised approach. In *DASFAA*, 2021.
[53] W. Zeng, X. Zhao, J. Tang, X. Lin, and P. Groth. Reinforcement learning-based collective entity alignment with adaptive features. *ACM Transactions on Information Systems*, 2021.
[54] W. Zeng, X. Zhao, W. Wang, J. Tang, and Z. Tan. Degree-aware alignment for entities in tail. In *SIGIR*, 2020.
[55] Q. Zhang, Z. Sun, W. Hu, M. Chen, L. Guo, and Y. Qu. Multi-view knowledge graph embedding for entity alignment. In *IJCAI*, 2019.
[56] R. Zhang, B. D. Trisedya, M. Li, Y. Jiang, and J. Qi. A comprehensive survey on knowledge graph entity alignment via representation learning. *CoRR*, abs/2103.15059, 2021.
[57] X. Zhao, W. Zeng, J. Tang, W. Wang, and F. Suchanek. An experimental study of state-of-the-art entity alignment approaches. *IEEE Transactions on Knowledge and Data Engineering*, 2020.
[58] H. Zhu, R. Xie, Z. Liu, and M. Sun. Iterative entity alignment via joint knowledge embeddings. In *IJCAI*, 2017.

Zequn Sun is currently a PhD student at Department of Computer Science and Technology, Nanjing University, China. He received his BS degree in Computer Science and Technology in 2016 from Hohai University, China. His research interests include knowledge graph representation learning and its applications such as entity alignment, link prediction and type inference.

Wei Hu is an associate professor at State Key Laboratory for Novel Software Technology, Department of Computer Science and Technology, Nanjing University, Nanjing, China. He received his PhD degree in Computer Software and Theory in 2009, and BS degree in Computer Science and Technology in 2005, both from Southeast University, China. His main research interests include knowledge graph, data integration and intelligent software.

Chengming Wang received his MS degree in Computer Science and Technology in 2018 from Nanjing University, China. He received his BS degree in Digital Media Technology in 2018 from Nanjing University of Posts and Telecommunications, China. His research interests include graph neural networks and entity alignment.



Yuxin Wang is currently a master student at Department of Computer Science and Technology, Nanjing University, China. She received her BS degree in Computer Science and Technology in 2018 from Nanjing University, China. Her research interests include knowledge graph embedding and incremental learning.

Yuzhong Qu is a full professor at State Key Laboratory for Novel Software Technology, Department of Computer Science and Technology, Nanjing University, China. He got the PhD degree in Computer Software in 1995 from Nanjing University, China, MS degree in Mathematics in 1988 from Fudan University, China, and BS degree in Mathematics in 1985 from Fudan University, China. His research interests include Semantic Web, question answering, and novel software technology for the Web.