Semantics Driven Embedding Learning for Effective Entity Alignment

[Technical Report]

Ziyue Zhong[†], Meihui Zhang[†*], Ju Fan[‡], Chenxiao Dou[§]
[†]*Beijing Institute of Technology*, [‡]*Renmin University of China*, [§]*KE Holdings Inc*
ziyue_zhong@bit.edu.cn, meihui_zhang@bit.edu.cn, fanj@ruc.edu.cn, douchenxiao001@ke.com

***Abstract*—Knowledge-based data service has become an emerging form of service in the world wide web (WWW). To ensure the service quality, a comprehensive knowledge base has to be constructed. Knowledge base integration is often a primary way to improve the completeness. In this paper, we focus on the fundamental problem in knowledge base integration, i.e., entity alignment (EA). EA has been studied for years. Traditional approaches focus on the symbolic features of entities and propose various similarity measures to identify equivalent entities. With recent development in knowledge graph representation learning, embedding-based entity alignment has emerged, which encodes the entities into vectors according to the semantic or structural information and computes the relatedness of entities based on the vector representation. While embedding-based approaches achieve promising results, we identify some important information that are not well exploited in existing works: 1) The neighboring entities contribute differently in the EA process, and should be carefully assigned the importance in learning the relatedness of entities; 2) The attribute values (especially the long texts) contain rich semantics that can build supplementary associations between entities.**

To this end, we propose SDEA - a Semantics Driven entity embedding method for Entity Alignment. SDEA consists of two modules, namely attribute embedding and relation embedding. The attribute embedding captures the semantic information from attribute values with a pre-trained transformer-based language model. The relation embedding selectively aggregates the semantic information from neighbors using a GRU model equipped with an attention mechanism. Both attribute embedding and relation embedding are driven by semantics, building bridges between entities. Experimental results show that our method significantly outperforms the state-of-the-art approaches on three benchmarks.

***Index Terms*—Entity Alignment, Semantics Driven, Transformer, Knowledge Base Integration**

I. Introduction

Knowledge-based data service has become an emerging and essential form of service in recent years. Many applications, including knowledge-based question answering [1], semantic search [2], and recommender systems [3], leverage knowledge bases to improve their overall service quality and user satisfaction.

The underlying knowledge base (KB), as the foundation of the knowledge-based service, needs to be well-constructed. There are a growing number of large-scale knowledge bases available on the web, such as YAGO [4], DBpedia [5], Freebase [6], and IMDb [7], in which the data are complementary and partially duplicated. A primary way to obtaining a more comprehensive knowledge base is to perform an integration process over various knowledge bases.

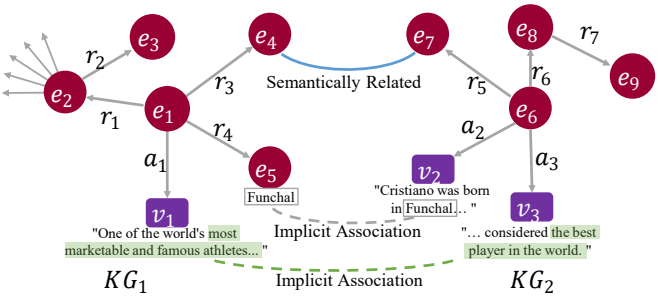


Fig. 1. Features in knowledge graph, where e, r, a, v represent entities, relations, attributes, and values respectively. In addition to the direct relations between entities (gray arrows), the information in textual values also indicates implicit associations (dashed lines), either directly through the attribute values, or indirectly through the entities' neighbors. e_2 represents a general concept, which has a large number of neighboring entities.

Entity alignment, as a major step of knowledge base integration, has been exploited extensively in the past and a variety of methods have been proposed. With recent advances in the field of knowledge graph representation learning, embedding-based methods become popular in the problem of entity alignment for knowledge graphs (KG, the major form of KB), mainly because they are free from expert participation and achieve competitive performance. These methods commonly divide entity alignment pipeline into two modules: entity embedding and entity alignment (based on the embeddings) [8]. The entity embedding module encodes entities into a vector space, whereas the entity alignment module captures the correspondence of embedding vectors with seed alignment as training data. To encode the entities, most of current research works commonly use two types of embedding: *relation embedding* and *attribute embedding*.

Relation embedding methods capture the structural information. The first type of methods exploit the relational association (e.g., r_i in Fig. 1) between entities. Some of them interpret a relation as the translation from one entity to another and learn entity and relation embeddings together [9]–[14]. Others use

* contact author

multi-hot vector to express relation names and learn part of the entity embeddings from relation vectors [15]. The second type of methods use graph neural networks (e.g., GCNs [16] or GATs [17]) to learn entity embeddings from the topological connections [15], [18]–[22]. The third type of methods exploit the long-term relational dependencies [14], [23]. They learn entity embeddings by transmitting information through the relational paths, i.e., a set of nose-to-tail linked triples (e.g., $(e_1, r_1, e_2), (e_2, r_2, e_3)$ in Fig. 1).

Attribute embedding methods capture the entities' additional information from attributes. The first type of methods exploit the correlations among attributes. Some of them use multi-hot vector to encode attribute names (e.g., a_i Fig. 1) and learn part of the entity embeddings from attribute vectors [15], [18]. Others use Skip-gram [24] to learn attribute embeddings from attribute name correlations and take the average of attribute embeddings as part of the entity representations [10]. The second type of methods learn initial or part of the entity embeddings from entity names/descriptions [15], [19], [20], [25].

While embedding-based entity alignment has shown its promise via joint learning from relations and attributes, we notice that some important information in KG is neglected or not well used. From a careful analysis on various KGs, we have the following key observations:

1) Some entities in KG, which are usually the ones representing general concepts like people, have a large number of neighboring entities (e.g., e_2 in Fig. 1). Such entities normally contribute less or even introduce noisy information in the entity alignment process. Further, when comparing an entity pair from two KGs, the semantically related neighbors (e.g., e_4 and e_7 in Fig. 1) are more helpful in either providing evidence for the alignment or identifying the conflicts for the unmatched pairs. Existing structure-based relation embedding methods do not well identify the contributions of the neighbours. Specifically, TransE-based [9]–[14] and GCN-based [15], [18]–[20] methods have no mechanism to distinguish the neighbors. GAT-based methods [21], [22] can distinguish the entity neighbors to some extent by learning weights from the graph structure. However, solely relying on the structure features is insufficient to identify the contributions as it ignores the semantics carried with the entities.

2) The attribute values (especially the long texts) usually contain rich information, which in many cases can build new associations between entities. On the one hand, the information inherent in the textual values can suggest implicit relationships between the entities (e.g., the new association between attribute value v_1 and v_3 in Fig. 1). On the other hand, the textual attributes may also indicate associations with the surrounding entities, implying indirect relationships through the neighbors (e.g., the new link from attribute value v_2 to entity e_5, and indirectly to e_1). According to our statistics, long textual attributes are not uncommon in KBs. For example, more than 15% of attributes contain long textual values (longer than 50 words) in Freebase, and more than 8% in DBpedia. We believe that such information is not only helpful in building bridges between entities, it is also highly reliable since it is drawn based on semantics. Nevertheless, joint learning from attributes and neighbor entities is still unexplored in existing works.

To well capture the semantics inherent in the entities (mostly in the long textual attribute values) and effectively identify the contribution of the neighbors in the alignment process, in this paper, we propose SDEA – a **S**emantic-**D**riven entity embedding method for **E**ntity **A**lignment. We employ two embedding modules, namely attribute embedding and relation embedding. In the attribute embedding module, a transformer-based pre-trained language model (or transformer for short) [26] is used to get initial embeddings of each entity. On top of that, the module captures the fine-grained semantics and the direct associations of entities. Further, the relation embedding module learns the contribution of neighboring entities from the fine-grained semantics of their attribute embeddings through attention mechanism and selectively aggregates the information from neighbors. Additionally, a joint representation learning is performed from the attribute embedding (holding the semantics of the entity) and the relation embedding (aggregating the semantics from the neighbors) to discover the indirect associations between entities.

To summarize, our contributions are summarized as follows:

- We identify that the semantics embedded with the entities play a crucial role in discovering the relatedness of entities, and propose a novel semantics-driven entity embedding method for entity alignment in knowledge bases.
- We design an effective learning framework equipped with an attribute embedding module and a relation embedding module that is capable of capturing the hidden semantics from the entity attributes and discerning the importance of the neighbors.
- We conduct extensive experiments to evaluate the effectiveness of our approach on three benchmark datasets: DBP15K [10], SRPRS [23], and OpenEA [8]. The experimental results show that we significantly outperform the state-of-the-art approaches.

The rest of the paper is organized as follows. Section II defines the problem of entity alignment and discusses the design considerations of our method. Section III describes the proposed entity alignment framework. Section IV presents the implementation details. Section V presents the experimental results. Section VI discusses related works and finally Section VII concludes the paper.

II. Definitions and Solution Design

In this section, we first introduce the formal definitions of knowledge graph (KG) and entity alignment (EA). Then we discuss the design considerations of our method.

A. Problem Definition

Definition 1 (Knowledge Graph)**.** A knowledge graph (KG) is denoted as $KG = \{E, R, A, V, T_r, T_a\}$. $e_i \in E$, $r_j \in R$, $a_k \in A$, and $v_l \in V$ represents an entity, a relation, an attribute, and

an attribute value respectively. $(e_i, r_j, e_k) \in T_r(e_i)$ denotes a relational triple, and $(e_i, a_j, v_k) \in T_a(e_i)$ denotes an attributed triple. Relational triples can also be represented as (h, r, t), where h is called *head entity* and t is called *tail entity*.

Definition 2 (Entity Alignment in KG)**.** Given two KGs (KG_1 and KG_2) and a function E that maps an entity to its real-world object, the goal of entity alignment is to find all matching pairs such that:

$$\hat{P} \subset KG_1 \times KG_2, (e_i, e_j) \in \hat{P} \iff E(e_i) = E(e_j) \quad (1)$$

where (e_i, e_j) is a matching pair and $\hat{P}$ is the set of the matching pairs. Usually, the left entity of the equivalent entity pair (i.e., e_i) is called *source entity* and the right entity of the equivalent entity pair (i.e., e_j) is called *target entity*.

As our paper does not assume one-to-one alignment across KGs, for each source entity, the target entities are ranked with the similarity score calculated by the alignment model. The higher score indicates the more likelihood that the target entity is equivalent to the source entity.

Remarks. Note that the current version of SDEA targets binary relations which are adopted in most existing KGs, such as YAGO [4], DBpedia [5] and Freebase [6]. We will explore more interesting directions, such as n-ary relations and richer KB structures in the future work.

B. Design Considerations

1) Identifying the Contribution of Neighbors: It is important to identify the contribution of neighboring entities in entity alignment. There are two aspects of reasons we have observed from real-world KGs, which motivate our design:

a) *Concept granularity*. Regarding the granularity, neighboring entities can be broadly categorized into two types: entities representing general concepts and entities representing specific concepts. Intuitively, specific-concept entities should contribute more than general-concept entities in entity alignment. Specific-concept neighbours can provide fine-grained information from multiple dimensions, whereas general-concept entities only provide abstract information or a broad category information. Further, general-concept entities normally link to a large number of entities (e.g., ⟨person⟩ in YAGO3 link to 2,231,431 entities). As such, transmitting information from general-concept entities may introduce noise from irrelevant entities. Take the two KGs KG_1 and KG_2 in Fig. 2 as an example. Consider the two blue entities $e_{1,1}$ ⟨C._Ronaldo⟩ and $e_{2,5}$ ⟨Cristano_Ronaldo⟩ to be aligned from the two KGs. Three of the surrounding entities of $e_{1,1}$ ⟨C._Ronaldo⟩, including $e_{1,3}$ ⟨C.D._Nacional⟩, $e_{1,4}$ ⟨Real_Madrid_C.F.⟩, and $e_{1,5}$ ⟨Academia_Sporting⟩, are specific-concept entities. These entities provide concrete information about the football clubs and the training facility, and thus they are discriminative and very useful in identifying who the person is. In contrast, $e_{1,2}$ ⟨Portugal⟩, $e_{1,6}$ ⟨player⟩ and $e_{1,7}$ ⟨person⟩ only provide the nationality and broad category information, which is of little use for aligning entities with the same type. Similarly, entities $e_{2,5}$ ⟨Cristano_Ronaldo⟩ has two neighbors with specific concepts and three with general concepts.

b) *Semantic relevance*. Considering the neighboring entities of a pair of to-be-aligned entities, we observe that the contribution of neighbors also differs with respect to whether a neighboring entity in one KG has a semantically related counterpart in the other KG. The neighbors with semantically related counterparts tend to contribute more in both aligning matching entities and identifying different entities. On the one hand, for entity pairs that are truly matched, their semantically related neighbors can provide direct extra evidence for the alignment. For example, when aligning $e_{1,1}$ ⟨C._Ronaldo⟩ and $e_{2,5}$ ⟨Cristano_Ronaldo⟩ in Fig. 2, there are four pairs of semantically related neighbors: $e_{1,7}$ ⟨person⟩ and $e_{2,4}$ ⟨people⟩ show that they are of the same type; $e_{1,2}$ ⟨Portugal⟩ and $e_{2,7}$ ⟨Portugal⟩ match them on the nationality; $e_{1,3}$ ⟨C.D._Nacional⟩ and $e_{2,6}$ ⟨C.D._Nacional⟩, $e_{1,4}$ ⟨Real_Madrid_C.F.⟩ and $e_{2,9}$ ⟨Real_Madrid_C.F._players⟩ indicate that both serve in the same teams. As such, the four pair of neighbors provide strong hints that ⟨C._Ronaldo⟩ and ⟨Cristano_Ronaldo⟩ are very likely to be the same person. In contrast, entities like $e_{1,5}$ ⟨Academia_Sporting⟩ and $e_{2,8}$ ⟨Madeira⟩ have no strongly related entities as matching evidence, and therefore, are unable to contribute much in the alignment. On the other hand, when comparing entity pairs that are not matched, the semantically related neighbors are helpful in identifying conflicting facts. Consider the example again. When comparing $e_{1,8}$ ⟨F.W._Bruskewitz⟩ and $e_{2,5}$ ⟨Cristano_Ronaldo⟩, we identify two pairs of semantically related neighbors: $e_{1,10}$ ⟨United_States⟩ and $e_{2,7}$ ⟨Portugal⟩ conflict on the nationality; $e_{1,9}$ ⟨Milwaukee⟩ and $e_{2,8}$ ⟨Madeira⟩ conflict on the place of birth. Therefore, the two entities are more likely to be different persons.

Taking concept granularity and semantic relevance into consideration, we believe that the neighboring entities should be assigned different importance concerning their contribution in the entity alignment process. Intuitively, neighbors carrying specific concepts and having strong related entities should be paid close attention (illustrated in dark red in Fig. 2). Contrarily, neighbors representing general concepts and with no related entities should be given low importance (illustrated in light red in Fig. 2). Both concept granularity and semantic relevance of the entities are regarding the semantic features. To well capture the semantics of the entities, which are mostly hidden in the textual attribute values, we design to use a transformer-based architecture that has been proved to be powerful in capturing the dependencies among words. With training data, the transformer model can be fine-tuned to pay more attention to the effective features. Further, to quantify the importance of neighboring entities, we adopt attention mechanism to learn the contribution of neighbors and selectively aggregate the information to represent the relations. As such, we learn an effective mapping between the semantic features and the neighbor contributions with training data.

2) Handling the Alignment of Long-tail Entities: Long-tail entities are the entities having few triples and far from seed

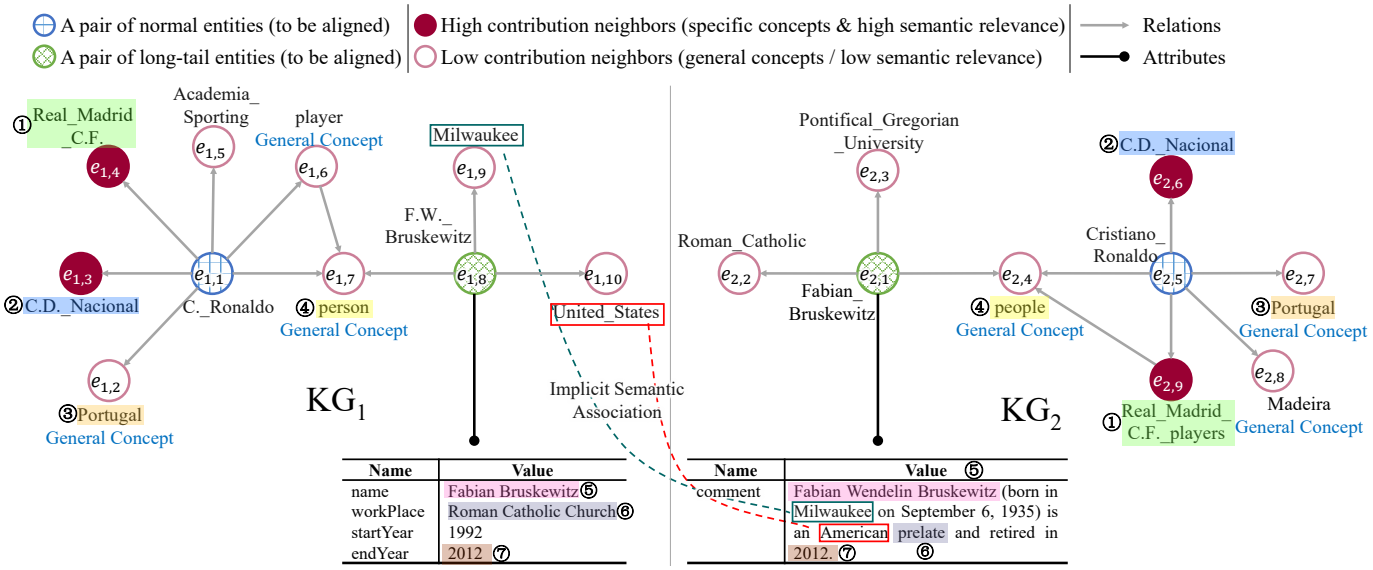


Fig. 2. An illustrating example with two pairs of to-be-aligned entities, where $e_{1,i}$ and $e_{2,j}$ denote the entities in KG_1 and KG_2 respectively, blue nodes represent normal entities and green nodes represent long-tail entities. The matching information is highlighted in the same background color and labeled with the same serial number. The attributes of $e_{1,1}$ and $e_{2,5}$ are omitted for the ease of presentation. Note that the overlapping of entity names is for better understanding and there is no assumption that entity names need to be consistent.

entities (the pre-aligned entities used as training data) [23]. Aligning long-tail entities is difficult because they have little information in the attributes and few neighbors with relations, which are inadequate to support the similarity learning in the alignment. Consider our real-world example in Fig. 2. The two green nodes $e_{1,8}$ ⟨F.W._Bruskewitz⟩ and $e_{2,1}$ ⟨Fabian_Bruskewitz⟩ represent the long-tail entities. Both have only 3 neighbors (vs. 6 and 5 neighbors for normal entities $e_{1,1}$ and $e_{2,5}$). Out of the three neighbors, they only match on very general entities, i.e., $e_{1,7}$ ⟨person⟩ and $e_{2,4}$ ⟨people⟩, which are of little use for alignment. Further, they do not contain directly related attributes. In fact, entity $e_{2,1}$ ⟨Fabian_Bruskewitz⟩ only has one single attribute comment with a long textual value. Simple similarity measures will not be able to detect the matching information hidden in the text, and as a result, fail to align the two entities. From a careful examination on real-word knowledge bases, we observe that in many cases long textual attributes (e.g., comment) contain rich and meaningful information, which is of great use in inferring associations between entities.

a) *Direct association through attributes*. We notice that in real-world KGs, more often than not, the to-be-aligned entity pairs do not have matching attributes. As in our example, none of the four attributes of $e_{1,8}$, i.e., name, workPlace, startYear, and endYear, directly matches with the attribute comment of $e_{2,1}$. However, a closer comparison reveals that the attribute values of truly matched entity pairs do agree on the underlying semantics to a large extent. As shown in the example, the comment text indeed contains information matched with the values of $e_{1,8}$ with high coherence. As a result, with a smart model capable of capturing the semantics embedded in the (long textual) attribute values, new associations can be built between the entities, making up for the scarce information of long-tail entities.

b) *Indirect association through neighbors*. Apart from the associated attributes, entities connect to a number of neighbors via relations. We observe that the long textual attributes are also helpful in relating entities through neighbors. In other words, it creates indirect associations between entities. Take the example in Fig. 2. Entity $e_{1,8}$ ⟨F.W._Bruskewitz⟩ has two neighbors, $e_{1,9}$ ⟨Milwaukee⟩ and $e_{1,10}$ ⟨United_States⟩, specifying his birthplace and nationality. Note that they have no matching information at entity level in KG_2. However, both information are mentioned in the long text attribute comment. Therefore, if we can identify the semantic association between comment and $e_{1,9}$ ($e_{1,10}$), an indirect link can be built between $e_{1,8}$ and $e_{2,1}$ through the neighbors.

The key to build the direct associations between entities is to uncover the fine-grained semantics inherent in the attributes, which is in line with our design for capturing the concept granularity and semantic relevance. We resort to transformer model to learn expressive representations, and the association can be derived from the the distance between the learnt representations of attributes. Further, to build the indirect association, we need to consider information from both attributes and neighbors jointly. To this end, we employ an additional MLP (Multi-Layer Perceptron) layer to learn a joint representation combining the attribute embedding of an entity with its relation representation aggregated from neighbors. In this way, we discover new associations for aligning long-tail entities.

III. Semantics-Driven Entity Embedding

Based on the aforementioned design considerations, we introduce SDEA – a **S**emantics-**D**riven entity embedding

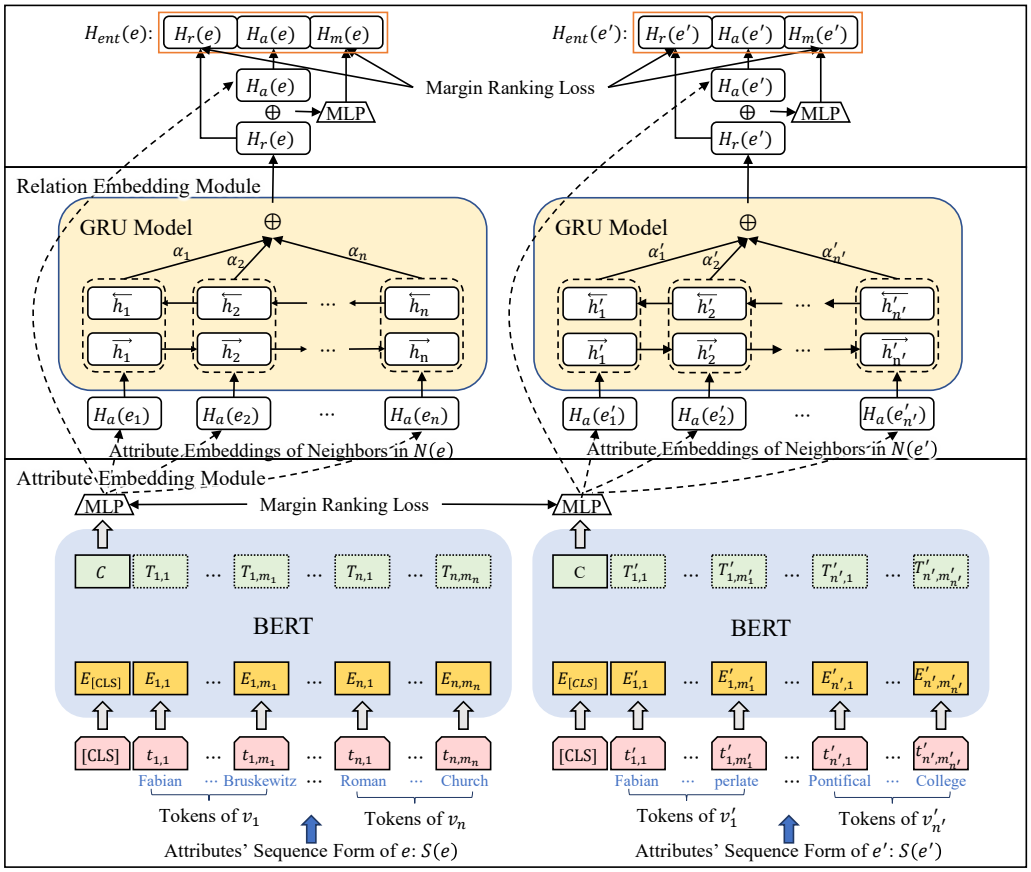


Fig. 3. The framework of proposed method. e and e' denote the entities in KG_1 and KG_2 respectively. $H_a(e)$, $H_r(e)$, and $H_m(e)$ denote the attribute embedding, relation embedding, and joint embedding of e respectively. $H_{ent}(e)$ denotes the final entity embedding.

framework for **E**ntity **A**lignment. As shown in Fig. 3, SDEA mainly consists of an *attribute embedding* module and a *relation embedding* module. The basic idea of SDEA is to generate an embedding-based representation $H_{ent}(e_i)$ for each entity e_i by utilizing the two modules.

- The attribute embedding module computes an attribute embedding, denoted by $H_a(e_i)$, by capturing semantic information in e_i's attribute values, especially long text.
- The relation embedding module computes a relation embedding, denoted by $H_r(e_i)$ by learning the contribution of neighbors and selectively aggregating the information.

In this section, we first present how to compute attribute embedding $H_a(e_i)$ and relation embedding $H_r(e_i)$ in Sections III-A and III-B respectively. Then, we discuss how to compute the joint representation $H_{ent}(e_i)$ based on $H_a(e_i)$ and $H_r(e_i)$ in Section III-C.

A. Attribute Embedding Module

The attribute embedding module aims to capture the semantics association between entities from their attribute values. From our observation, the attribute values consist of not only short texts and numbers (in structured fields), but also long sentences (in textual fields). Capturing the semantic information from attribute values has always been a challenge. On the one hand, conventional string similarity-based approaches fail to handle heterogeneity in the text, such as synonyms, polysemy words, abbreviations, and numbers with different precision and units. On the other hand, the alignment of attributes raises another challenge due to the heterogeneity of different KG schema. Due to these challenges, most existing studies on entity alignment avoid handling attribute values. Instead, they only consider to capture semantics from entity names or descriptions.

To address the problem, we propose to employ a transformer model [27], which has achieved the start-of-the-art performance in capturing the semantic information of texts, to handle the heterogeneity in the text. Further, to handle the heterogeneity of different KG schema, we capture the fine-grained semantics by combining all attribute values of an entity as a whole, and then capture the semantic associations between two entities. Our design consideration is that the to-be-aligned entity pairs sometimes do not have matching attributes, especially for the long-tail entities. In this case, the semantic associations cannot be found based on attribute alignment. One highlight of using transformer is that it pays more attention to the features that conveying the text semantics during fine-tuning. As a result, we do not need to pay special attention to the order of attribute values. Instead, we can simply arrange

Algorithm 1: KG transformation method

Input:
E: the entity set in KG
A: the attribute set in KG
T_a: the set of attributed triples in KG
Output:
S: the set of generated attribute sequences of entities in E

1 $\hat{O}(A) \leftarrow$ initialize a random order of A
2 **for** $e_i \in E$ **do**
3 **for** $a_j \in \hat{O}(A)$ **do**
4 **for** $(e_i, a_j, v_k) \in KG$ **do**
5 append (e_i, a_j, v_k) to $\hat{T}_a(e_i)$
6 **end**
7 **end**
8 **for** $(e_i, a_j, v_j) \in \hat{T}_a(e_i)$ **do**
9 append v_j to $\hat{V}(e_i)$
10 **end**
11 concatenate the values in $\hat{V}(e_i)$ to form $S(e_i)$
12 **end**

the attribute values of entities in each knowledge graph in the same order to form a contextual relationship between attribute values, which are then fed into the transformer model.

For ease of presentation, we take BERT [26], an representative model of transformer, as an example to describe our method as follows. We formalize attribute embedding as a downstream task of BERT, i.e., fine-tuning a pre-trained BERT to encode attribute values of entity e_i into attribute embedding $H_a(e_i)$ in the following two phases.

1) Data Preprocessing: This phase aims to transform attribute values of entity e_i into a *sequence*, i.e., a series of tokens, which can be then fed into our BERT model.

Algorithm 1 describes the overall procedure of this step. Recall that, in Definition 1, we define A as the set of attribute names in KG and $T_a(e_i)$ as the set of attributed triples for entity e_i (i.e., e_i is the head entity of all triples in $T_a(e_i)$). Initially, the algorithm generates an order of all attributes in A, denoted as $\hat{O}(A) = [a_1, a_2, ..., a_n]$, where $a_i \in A$ and $\{a_i\}|_{i=1}^{n} = A$ (line 1). Then, it organizes the triples in $T_a(e_i)$ in line with the order of $\hat{O}(A)$ (Line 3-7), i.e.,

$$\hat{T}_a(e_i) = [(e_i, \boldsymbol{a_1}, v_{k_1}), (e_i, \boldsymbol{a_2}, v_{k_2}), ..., (e_i, \boldsymbol{a_n}, v_{k_n})] \quad (2)$$

where $\hat{T}_a(e_i)$ for all entities follow the same order. Finally, the algorithm concatenates attribute values from triples in $\hat{T}_a(e_i)$ to form a sequence of tokens (Line 8-11), ie.,

$$\hat{V}(e_i) = [v_1, v_2, ..., v_n] \quad (3)$$

$$S(e_i) = \text{“}t_{1,1}\ ...\ t_{1,m_1}\ ...\ t_{n,1}\ ...\ t_{n,m_n}\text{”} \quad (4)$$

where $t_{i,1}, ..., t_{i,m_i}$ is the tokens of v_i. For example, Fig. 4 illustrates the procedure in Algorithm 1.

2) Attribute Encoding: This phase takes the sequence generated in the previous step as input, and aims to transform the sequence into an embedding via our BERT model.

To this end, we use a pre-trained BERT model and an MLP (Multi-Layer Perceptron) layer to obtain an attribute

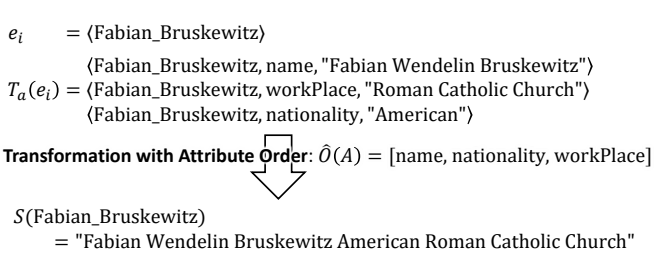


Fig. 4. An example of the transformation procedure.

embedding vector $H_a(e_i)$ by encoding the attribute sequence $S(e_i)$ of entity e_i. The process are divided into the following three steps: 1) the first step adds the special token “[CLS]” in the beginning of sequence $S(e_i)$, so as to meet the input requirement of BERT, and results in an input sequence “[CLS]” || $S(e_i)$ (e.g., “[CLS] Fabian Wendelin Bruskewitz American Roman Catholic Church”), denoted as $S'(e_i)$; 2) The second step feeds sequence $S'(e_i)$ into BERT and takes the final state, denoted by $C(e_i)$, corresponding to “[CLS]” as the intermediate vector representation of $S(e_i)$ [26]; 3) The third step further adds an MLP layer over $C(e_i)$to obtain the final attribute embedding of e_i. Formally, we present the above three steps as follows.

$$S'(e_i) = \text{“[CLS]” } || \ S(e_i) \quad (5)$$

$$C(e_i) = \text{BERT}(S'(e_i)) \quad (6)$$

$$H_a(e_i) = \text{MLP}(C(e_i)) \quad (7)$$

Remarks. As BERT uses a *subword-based tokenization* strategy to deal with rare words, it may not work well for numeric values [28]. There are two directions to address this problem: 1) handling the numeric values separately, and 2) exploring other pre-trained models, e.g., ELMo [29]. [28] has reported that ELMo has the best performance in capturing numeracy among all pre-trained methods. We are very interested in exploring this in the future.

Moreover, domain-specific terms may not be handled well by BERT. This can be partially addressed by extending pre-trained models with domain-specific vocabulary by pre-training on the domain-specific corpus [30], [31]. We will study this problem in the future work.

B. Relation Embedding Module

In relation embedding module, we use a GRU-based attention mechanism [32] to model contribution of neighbors and selectively aggregate this information. Intuitively, given an entity e_i, the contribution of its neighbor e_j would depend on other neighbors of e_i. Alternative methods include averaging the neighbor's embeddings, pooling, and directly using the attention mechanism. Compared to them, the Bi-directional GRU (BiGRU) [33] model is able to capture correlations among different neighbors of e_i, thus it can model different contributions of the same neighbor for different entities according to the context information (the surrounding neighbors). More specifically, the BiGRU model generates a

set of entity-specific neighbor embeddings, i.e., $\{h_t\}|_{t=1}^{n}$. The correlations are captured by the *reset gate* and the *update gate* of BiGRU, which are used to control how much information should be dropped from previous hidden states and how much information should be retained from current input. Further, the attention mechanism uses the correlations to compute the contributions and selectively aggregate the neighbor embeddings. Next, we introduce the BiGRU model and the attention mechanism as follows.

1) Capturing Correlations among Neighbors with BiGRU: Intuitively, we take all neighbors of entity e_i as an input sequence of our BiGRU model. Given entity e_i, let x_t denote the t-th input embedding (the attribute embedding of e_i's t-th neighbor) and let h_t denote the output vector of the t-th hidden unit. The reset gate r_t drops the neighbor information which is introduced previously but not important in determining the correlations, i.e.,

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \tag{8}$$

$$\tilde{h}_t = \phi(W x_t) + U(r_t \odot h_{t-1} + b_h) \tag{9}$$

where W, U, and b are parameter matrices and vector $\tilde{h}_t$ is the hidden state of t-th unit, σ is the sigmoid function, ϕ is the hyperbolic tangent, and $\odot$ denotes the *Hadamard product* [1]. Next, an update gate z_t is introduced to include the important features from the current input neighbor x_t:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \tag{10}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{11}$$

We adopt the bidirectional version of GRU model, in which $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$ are the t-th output for each direction. The final output h_t of t-th hidden unit is the sum of $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$.

2) Attention Mechanism: The attention mechanism enables the model to exploit importance of each neighbor to extract the most relevant information from the neighbors. The attention mechanism first generates a global attention representation $\hat{h}$ for each entity and obtains the contribution of each neighbor by a scoring function. Specifically, given entity e_i and entity-specific neighbor embeddings $\{h_t\}|_{t=1}^{n}$, the global attention representation is learned from the last output representation of BiGRU model (i.e., h_n), because the aggregated input information containing in h_n can help the model identify the important parts. In our design, $\hat{h}$ is obtained by sending h_n into an MLP layer:

$$\hat{h} = \mathrm{MLP}(h_n) \tag{12}$$

Further, the model computes the contribution of neighbors by a scoring function $f(\cdot, \cdot)$. The scoring function is used to capture the similarity between the global attention representation $\hat{h}$ and each neighbor embedding h_t. We use the inner product function in our design. The contributions are denoted as:

$$w_t = f(h_t, \hat{h}) = h_t^T \cdot \hat{h} \tag{13}$$

$$\alpha_t = \frac{\exp(w_t)}{\sum_{i=1}^{n} \exp(w_i)} \tag{14}$$

[1]https://en.wikipedia.org/wiki/Hadamard_product_(matrices)

where α_t weights the neighbors according to their contributions. Finally, our model generates the relation embedding for entity e_i as the weighted sum of its entity-specific neighbor embedding h_t:

$$H_r(e_i) = \sum_{t=1}^{n} \alpha_t \cdot h_t \tag{15}$$

C. Joint Entity Representation

Given an entity e_i, the attribute embedding module and the relation embedding module compute its attribute embedding $H_a(e_i)$ and relation embedding $H_r(e_i)$, capturing the information from attributes and neighbors respectively. To jointly model the information from attributes and neighbors, we compute a joint representation $H_m(e_i)$, which combines $H_a(e_i)$ and $H_r(e_i)$ with an MLP layer:

$$H_m(e_i) = \mathrm{MLP}([H_a(e_i); H_r(e_i)]) \tag{16}$$

where $[;]$ represents the concatenation of vectors. As such, we have three embeddings $H_a(e_i)$, $H_r(e_i)$, and $H_m(e_i)$, capturing the attribute information, the neighbor information, and the joint attribute and neighbor information, respectively. The final entity embedding $H_{ent}(e_i)$ is the concatenation of these embeddings, which captures all three aspects of information:

$$H_{ent}(e_i) = [H_r(e_i); H_a(e_i); H_m(e_i)] \tag{17}$$

Therefore, based on the joint embeddings, the distance between entity embeddings naturally measures the semantic relevance of neighbors, direct associations of attributes and indirect associations of attributes and neighbors.

IV. Implementation

Given the rationales and the architecture of our proposed SDEA framework presented previously, in this section, we focus on the implementation details.

A. Model Training

A simple method to train our attribute embedding and relation embedding modules is to connect these modules in an end-to-end architecture and train them together. However, considering the limitation of GPU memory and the running efficiency, we separate the training of the attribute embedding module in our implementation, because fine-tuning the transformer model consumes much GPU memory. For ease of presentation, we use $H(\cdot)$ and $H'(\cdot)$ to represent embeddings of entities from KG_1 and KG_2 respectively.

1) Pre-training the Attribute Embedding Module: The pre-training procedure of attribute embedding module is shown in Algorithm 2. The algorithm takes as input the entity sets E and E' of the two KGs (KG_1 and KG_1) with their corresponding attribute triples, the training set $\hat{L}_{train}$, and validation set $\hat{L}_{valid}$ of aligned entities. The output of the algorithm is a trained attribute embedding module, which is denoted by $\mathrm{AttrModule}(\cdot)$.

The pre-training procedure in Algorithm 2 consists of max_epoch epochs, where each epoch has the following

Algorithm 2: Pre-training of attribute embedding module

Input:
E, E': two sets of entities to be aligned from two KGs.
$\hat{L}_{train}$, $\hat{L}_{valid}$: the training set and validation set.

1 **for** *k in range(max_epoch)* **do**
2 $H_a \leftarrow$ calculate initial embedding $H_a(e_i)$ for each entity $e_i \in E$ by using $H_a(e_i) = \text{AttrModule}(e_i)$
3 $H'_a \leftarrow$ calculate initial embedding $H'_a(e'_j)$ for each entity $e'_j \in E'$ by using $H'_a(e'_j) = \text{AttrModule}(e'_j)$
4 $M^a_{candidates} = \text{GenCandidates}(H_a, H'_a)$
5 **for** $(e_i, e'_i) \in \hat{L}_{train}$ **do**
6 $e''_i \in M^a_{candidates}(e_i) \wedge e''_i \neq e'_i$
7 $H_a(e_i), H'_a(e'_i), H'_a(e''_i) \leftarrow \text{AttrModule}(e_i/e'_i/e''_i)$
8 $Loss_i = \text{MarginLoss}(H_a(e_i), H'_a(e'_i), H'_a(e''_i))$
9 Update parameters of AttrModule via back-propagating $Loss_i$
10 **end**
11 validate AttrModule using validation set $\hat{L}_{valid}$
12 **end**

Algorithm 3: Model training with pre-trained attribute embeddings

Input:
E, E': two sets of entities to be aligned from two KGs.
$\hat{L}_{train}$, $\hat{L}_{valid}$: the training set and validation set.
H_a, H'_a: the pre-trained attribute embeddings.

1 $M^r_{candidates} = \text{GenCandidates}(H_a, H'_a)$
2 **for** *k in range(max_epoch)* **do**
3 **for** $(e_i, e'_i) \in \hat{L}_{train}$ **do**
4 $e''_i \in M^r_{candidates}(e_i) \wedge e''_i \neq e'_i$
5 $H_r(e_i), H'_r(e'_i), H'_r(e''_i) \leftarrow \text{RelModule}(e_i/e'_i/e''_i)$
6 $H_m(e_i) = \text{MLP}([H_a(e_i); H_r(e_i)])$
7 $H'_m(e'_i) = \text{MLP}([H'_a(e'_i); H'_r(e'_i)])$
8 $H'_m(e''_i) = \text{MLP}([H'_a(e''_i); H'_r(e''_i)])$
9 $Loss_i = \text{MarginLoss}([H_r(e_i); H_m(e_i)], [H'_r(e'_i); H'_m(e'_i)]), [H'_r(e''_i); H'_m(e''_i)])$
10 Update parameters of RelModel and MLP layer via back-propagating $Loss_i$
11 **end**
12 validate the model with $\hat{L}_{valid}$
13 **end**

three steps. The first step (Line 2-4) retrieves a set of candidates $M^a_{candidates}(e_i)$ from KG_2, given each each entity e_i in KG_1. To this end, it computes pairwise similarities for pairs of entities across the two KGs on top of the initial entity embeddings produced by AttrModule. The second step (Line 5-10) updates parameters of neural networks using the input training set $\hat{L}_{train}$. Specifically, for each entity pair in training set (denoted as (e_i, e'_i)), we first randomly select a negative sample e''_i, which is not matched with e_i, from candidate set $M^a_{candidates}(e_i)$ of e_i. Then, we feed the entities e_i, e'_i and e''_i into our attribute embedding module to generate attribute embeddings $H_a(e_i)$, $H'_a(e'_i)$, and $H'_a(e''_i)$ respectively. These embeddings are then used to calculates the loss, and a back propagation process is performed to update parameters of the attribute embedding module based on the loss. In particular, we use the following margin-based ranking loss (Line 8), as this loss function has successful practices in binary classification and is commonly adopted in entity alignment approaches. Compared to other alternative loss functions used in binary classification, such as cross-entropy loss, margin-based ranking loss can make the distance between embeddings of unmatching entities large and provide better generalization ability.

$$\mathcal{L} = \sum_{e_i, e'_i, e''_i \in D} \max\{0, \rho(H_a(e_i), H'_a(e'_i)) - \rho(H_a(e_i), H'_a(e''_i)) + \beta\} \tag{18}$$

where D is our training set with the generated negative samples, ρ is the l_2 distance function, and $\beta > 0$ is the margin hyper-parameter used for separating positive and negative pairs. At last, the third step is responsible for validating the module using validation set $\hat{L}_{valid}$ (Line 11). We omit the details of this step since the validation process is similar to the second (training) step.

2) Model Training with Pre-trained Attribute Embeddings: The training procedure is shown in Algorithm 3. Similar to the pre-training procedure, the algorithm also takes as input entity sets E and E' from the two KGs with their corresponding triples, training set $\hat{L}_{train}$, validation set $\hat{L}_{valid}$. Besides, it also considers a new input, namely the pre-trained attribute embeddings H_a for E and H'_a for E'. Based on the pre-trained entity embeddings, the algorithms first retrieves a fixed number of candidates $M^r_{candidates}(e_i)$ from KG_2 for each entity e_i in KG_1 (Line 1). Then, for each pair of training data (e_i, e'_i), it generates a negative sample in the same way of the above pre-training procedure. Next, the algorithm trains the models by generating the relation embeddings $H_r(e_i)$, $H'_r(e'_i)$, and $H'_r(e''_i)$ (Line 5), and the joint representations $H_m(e_i)$, $H'_m(e'_i)$, and $H'_m(e''_i)$ with MLP layers (Line 6-8), and then updating the model parameters with the margin-based ranking loss (Line 9-10). The loss is computed with the concatenation of the relation embeddings and the joint representations. Similarly as before, we validate the module using validation set $\hat{L}_{valid}$ (Line 12).

B. Entity Alignment

Entity alignment aims to find aligned entities between two KGs. Similar to other approaches [10]–[13], [21], [23], [34], we directly calculate the pairwise similarity score of entities on top of their final embeddings (i.e., $H_{ent}(e)$ and $H'_{ent}(e')$ in Fig. 3) via the cosine similarity function. The rationale of using the cosine similarity function is that there is a monotonic relationship between Euclidean distance (the l_2 distance used in our loss function) and the cosine similarity when the norm of the vector is normalized. Note that we do not assume that every entity in KB_1 must have (at most) a matching entity in KB_2. Thus, for each source entity in KG_1, we retrieve a list of target entities from KG_2 with cosine similarity scores, where high scores imply high matching probabilities.

TABLE I
STATISTICS OF BENCHMARKS

Datasets		Entities	Rel.	Attr.	Rel. triples	Attr. Triples
DBP15K						
ZH-EN	ZH	19,388	1,701	7,780	70,414	379,684
	EN	19,572	1,323	6,933	95,142	567,755
JA-EN	JA	19,814	1,299	5,681	77,214	354,619
	EN	19,780	1,153	5,850	93,484	497,230
FR-EN	FR	19,661	903	4,431	105,998	528,665
	EN	19,993	1,208	6,161	115,722	576,543
SRPRS						
EN-FR	EN	15,000	221	274	36,508	70,750
	FR	15,000	177	393	33,532	56,344
EN-DE	EN	15,000	222	275	38,363	62,715
	DE	15,000	120	185	37,377	142,506
DBP-WD	DBP	15,000	253	336	38,421	71,957
	WD	15,000	144	412	40,159	136,315
DBP-YG	DBP	15,000	223	300	33,748	69,355
	YG	15,000	30	21	36,569	22,519
OpenEA						
D_W_15K_V1	D	15,000	248	342	38,265	68,258
	W	15,000	169	649	42,746	138,246
D_W_100K_V1	D	100,000	413	493	293,990	451,011
	W	100,000	261	874	251,708	687,787

DBP (D), YG, WD (W) stand for DBpedia, YAGO, and Wikidata.

V. EXPERIMENTS

We conduct an extensive experimental study to evaluate our model and compare with the state-of-the-art baselines. In this section, we first present the experimental setup and then report the results and analysis.

A. Experimental Setup

1) Datasets: We conduct experiments on two widely-used benchmarks, DBP15K [10] and SRPRS [23], and two challenging datasets in OpenEA [8]. Table I provides more details of these datasets, including number of entities, relations, attributes, relation triples and attribute triples. We also examine the relation density in DBP15K, SRPRS, and OpeEA, and plot the degree distribution in Fig. 5. We can see that SRPRS and OpenEA contain more long-tail entities (with low degree), which are more closer to the real-world knowledge bases.

DBP15K [10] contains three multilingual datasets extracted from DBpedia, including Chinese-English (ZH-EN), Japanese-English (JA-EN), and Franch-English (FR-EN). Each dataset contains 15,000 inter-language links (ground truth) for training and testing. DBP15K has two versions, namely full version and condensed version [10], where the condensed version samples the relational triples with popular head and tail entities from the full version. To be consistent with the existing studies [10], [35], we use the condensed version to evaluate our proposal.

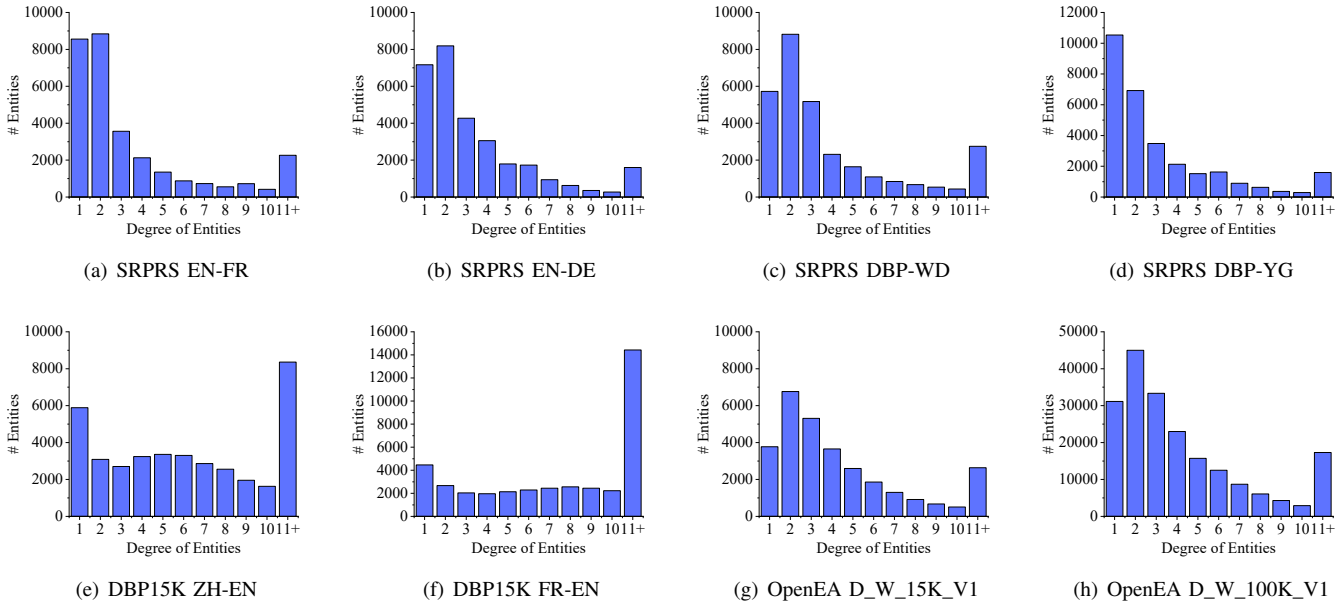
SRPRS [23] is a well-adopted benchmark for entity alignment, containing two multilingual datasets, EN-DE and EN-FR, and two monolingual datasets, DBP-WD and DBP-YG. More specifically, the multilingual datasets EN-DE and EN-FR are extracted from the multilingual DBpedia, while the monolingual datasets DBP-WD and DBP-YG are extracted from DBpedia, Wikipedia, and YAGO. Consistent with [35], we replace the entity identifiers in WD with entity names retrieved from Wikidata. The original SRPRS benchmark [23] has two versions, normal version and dense version. We follow a recent benchmarking study work [35] to use the normal version, as this version is closer to real-world knowledge bases. In particular, SRPRS contains 15,000 pair of equivalent entity links (ground truth) for training and testing. As shown in Table I, compared to DBP15K, SRPRS is more sparse (containing fewer relations).

OpenEA [8] is a newly proposed benchmark in 2020. It is similar to SRPRS, containing two multilingual (EN-DE and EN-FR) and two monolingual (D-W and D-Y) datasets with sparse (V1) and condensed (V2) version. Differently, it also provides a large version with 100K matching entities. Due to its similarity with SRPRS, we only present the experimental results on two datasets: D_W_15K_V1 and D_W_100K_V1, which are more challenging for two reasons: 1) they are sparse in relations, and 2) they contain limited information for entity matching. In particular, they do not well match on entity names, i.e., D (DBpedia) contains names (e.g., Poland), while W (Wikidata) uses Wikidata IDs (e.g., Q36).

2) Evaluation Metrics: We use Hits@K (K=1, 10) and mean reciprocal rank (MRR) as the evaluation metrics, which are described as follows. (1) Hits@K (or H@K for short) is defined as the proportion of source entities whose target entities are in the top-K matching results returned by an approach. The higher the Hits@K is, the better an approach is. (2) MRR ($MRR = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{rank_i}$) is defined as, given a set of source entities, the average of the reciprocal ranks of their ground-truth target entities in their matching results returned by an approach. Similar to Hits@K, the higher the MRR is, the better an approach is.

3) Experimental Settings: For each dataset, we split the ground truth links into training, validation, and test set with ratio of 2:1:7. We implement our method with Pytorch [36] and Transformers library [37]. In all experiments, we fix the max length of BERT's input sequence to be 128. The batch size is 8 for the attribute embedding module and 256 for the relation embedding module. The training process is terminated when Hits@1 on the validation set does not increase for 5 consecutive times, and then it returns the checkpoint with the best Hits@1 on the validation set.

4) Compared Methods: Zhao et al. [35] have conducted a comprehensive experimental study on the state-of-the-art methods (up to 2020) and reported the results on DBP15K and SRPRS benchmarks. Thanks to this work, we can evaluate our method comprehensively. In addition, we include a recent solution, namely BERT-INT [34], which also uses language model and achieves good performance on DBP15K. In Table II, we summarized the methods by considering the techniques used from different aspects, such as relational association, topological connection, etc. Note that since entity descriptions are not available in all benchmarks we used, HMAN only leverages GCNs and FNNs to capture topological connections, relational associations, and attribute correlations (same as the experiments in [35]), and BERT-INT uses entity names as an alternative. Further, CEA performs a stable matching algorithm for 1-1 alignment after embedding. For a fair comparison, we also evaluate the embedding only version of CEA, marked as CEA (Emb) in the tables.



(a) SRPRS EN-FR (b) SRPRS EN-DE (c) SRPRS DBP-WD (d) SRPRS DBP-YG

(e) DBP15K ZH-EN (f) DBP15K FR-EN (g) OpenEA D_W_15K_V1 (h) OpenEA D_W_100K_V1

Fig. 5. Entities' degree distribution of datasets in SRPRS and DBP15K.

TABLE II
BASELINE METHODS COMPARISON BASED ON THE TECHNIQUES USED

Method	Relational Association	Topological Connection	Long-term Dependency	Attribute Correlation	Literal
MTransE [9]	TransE				
JAPE-Stru [10]	TransE				
JAPE [10]	TransE			Skip-gram	
NAEA [11]	TransE variant				
BootEA [12]	TransE variant				
TransEdge [13]	TransE variant				
IPTransE [14]	TransE		PTransE		
RSN4EA [23]			RSNs		
GCN [18]		GCNs			
GCN-Align [18]		GCNs		GCNs	
MuGNN [21]	GATs	GATs			
KECG [22]	TransE	GATs			
HMAN [15]	FNNs	GCNs		FNNs	
RDGCN [19]	head/tail+GATs	GCNs+ Highway gates			Glove (Entity Name)
HGCN [20]	head/tail	GCNs+ Highway gates			Glove (Entity Name)
CEA [38]		GCNs			Levenshtein+ fastTest/MUSE (Entity Name)
BERT-INT [34]	BERT-based (Neighbor's Name)				BERT-based (Entity Name +Attribute Value)

TABLE III
EXPERIMENTAL RESULTS ON DBP15K BENCHMARK

Method	ZH-EN			JA-EN			FR-EN		
	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR
MTransE [9]	20.9	51.2	0.31	25.0	57.2	0.36	24.7	57.7	0.36
JAPE-Stru [10]	37.2	68.9	0.48	32.9	63.8	0.43	29.3	61.7	0.40
JAPE [10]	41.4	74.1	0.53	36.5	69.5	0.48	31.8	66.8	0.44
NAEA [11]	38.5	63.5	0.47	35.3	61.3	0.44	30.8	59.6	0.40
BootEA [12]	61.4	84.1	0.69	57.3	82.9	0.66	58.5	84.5	0.68
TransEdge [13]	75.3	92.4	0.81	74.6	92.4	0.81	77.0	94.2	0.83
IPTransE [14]	33.2	64.5	0.43	29.0	59.5	0.39	24.5	56.8	0.35
RSN4EA [23]	58.0	81.1	0.66	57.4	79.9	0.65	61.2	84.1	0.69
GCN [18]	39.8	72.0	0.51	40.0	72.9	0.51	38.9	74.9	0.51
GCN-Align [18]	43.4	76.2	0.55	42.7	76.2	0.54	41.1	77.2	0.53
MuGNN [21]	47.0	83.5	0.59	48.3	85.6	0.61	49.1	86.7	0.62
KECG [22]	47.7	83.6	0.60	49.2	84.4	0.61	48.5	84.9	0.61
HMAN [15]	56.1	85.9	0.67	55.7	86.0	0.67	55.0	87.6	0.66
RDGCN [19]	69.7	84.2	0.75	76.3	89.7	0.81	87.3	95.0	0.90
HGCN [20]	70.8	84.0	0.76	75.8	88.9	0.81	88.8	95.9	0.91
CEA (Emb) [38]	71.9	85.4	0.77	78.5	90.5	0.83	92.8	98.1	0.95
CEA [38]	78.7			**86.3**			97.2		
BERT-INT [34]	81.4	83.7	0.82	80.6	83.5	0.82	**98.7**	99.2	**0.99**
SDEA	**87.0**	**96.6**	**0.91**	84.8	**95.2**	**0.89**	96.9	**99.5**	0.98
SDEA w/o rel.	84.8	94.9	0.89	79.0	90.2	0.83	96.4	99.3	0.98

B. Experimental Results

This section presents the experimental results. We first compare the overall performance of our method and the baseline solutions in Section V-B1, and then evaluate the efficiency and scalability in Section V-B2. Next, we report how our method performs on handling long-tail entities in Section V-B3. Further, we provide an ablation study on the two main components, attribute embedding and relation embedding, of our method in Section V-B4. Finally, Section V-B5 provides a case study to illustrate that our method is effective for identifying the contributions of neighbors.

1) Overall Results: Tables III, IV, and V report the overall results of our method (SDEA) with 17 baselines. In summary, our method achieves better or comparable performance comparing to the state-of-the-art baselines on DBP15K and SRPRS, and significantly outperforms them on the challenging datasets (i.e., D_W_15K_V1 and D_W_100K_V1). Next, we analyze the results of baselines by examining the techniques as summarized in Table II.

The first group are TransE-based methods [9]–[13]. They learn entity and relation embeddings by exploiting relational association from relation triples. MTransE, JAPE-Stru (the structured-only variant of JAPE), and NAEA directly apply TransE to exploit relational association between entities. Due to the limitation of TransE on capturing 1-N, N-1, and N-N relations between entities (e.g., an entity may have same relation with many entities), these methods only achieve

TABLE IV
EXPERIMENTAL RESULTS ON SRPRS BENCHMARK

Method	EN-FR			EN-DE			DBP-WD			DBP-YG		
	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR
MTransE [9]	21.3	44.7	0.29	10.7	24.8	0.16	18.8	38.2	0.26	19.6	40.1	0.27
JAPE-Stru [10]	24.1	53.3	0.34	30.2	57.8	0.40	21.0	48.5	0.30	21.5	51.6	0.32
JAPE [10]	24.1	54.4	0.34	26.8	54.7	0.36	21.2	50.2	0.31	19.3	50.0	0.30
NAEA [11]	17.7	41.6	0.26	30.7	53.5	0.39	18.2	42.9	0.26	19.5	45.1	0.28
BootEA [12]	36.5	64.9	0.46	50.3	73.2	0.58	38.4	66.7	0.48	38.1	65.1	0.47
TransEdge [13]	40.0	67.5	0.49	55.6	75.3	0.63	46.1	73.8	0.56	44.3	69.9	0.53
IPTransE [14]	12.4	30.1	0.18	13.5	31.6	0.20	10.1	26.2	0.16	10.3	26.0	0.16
RSN4EA [23]	35.0	63.6	0.44	48.4	72.9	0.57	39.1	66.3	0.48	39.3	66.5	0.49
GCN [18]	24.3	52.2	0.34	38.5	60.0	0.46	29.1	55.6	0.38	31.9	58.6	0.41
GCN-Align [18]	29.6	59.2	0.40	42.8	66.2	0.51	32.7	61.1	0.42	34.7	64.0	0.45
MuGNN [21]	13.1	34.2	0.20	24.5	43.1	0.31	15.1	36.6	0.22	17.5	38.1	0.24
KECG [22]	29.8	61.6	0.40	44.4	70.7	0.54	32.3	64.6	0.43	35.0	65.1	0.45
HMAN [15]	40.0	70.5	0.50	52.8	77.8	0.62	43.3	74.4	0.54	46.1	76.5	0.56
RDGCN [19]	67.2	76.7	0.71	77.9	88.6	0.82	97.4	99.4	0.98	99.0	99.7	0.99
HGCN [20]	67.0	77.0	0.71	76.3	86.3	0.80	98.9	99.9	0.99	99.1	99.7	0.99
CEA (Emb) [38]	93.3	97.4	0.95	94.5	98.0	0.96	99.9	**100.0**	**1.00**	99.9	**100.0**	**1.00**
CEA [38]	96.2			97.1			**100.0**			**100.0**		
BERT-INT [34]	**97.1**	97.5	**0.97**	**98.6**	98.8	**0.99**	99.6	99.7	**1.00**	**100.0**	**100.0**	**1.00**
SDEA	96.6	**98.6**	**0.97**	96.8	**98.9**	0.98	98.0	99.6	0.99	99.9	**100.0**	**1.00**
SDEA w/o rel.	95.6	97.7	0.96	95.7	98.1	0.97	97.9	99.5	0.99	99.9	100.0	1.00

inferior performance. In particular, we can see that JAPE-Stru and NAEA outperform MTransE on all the datasets. The reason is that JAPE-Stru and NAEA utilize the negative sampling technique in the training process while MTransE does not. The experimental results show that negative samples are effective in distinguishing the relations between entities [10]. Notice that the methods BootEA and TransEdge achieve better performance than other TransE-based approaches, which can be partially attributed to the semi-supervised learning strategy they used.

The second group captures the long-term dependencies from paths, which can also handle the alignment of long-tail entities to a certain extent. However, IPTransE can only model short distance paths while modeling long distance paths in RSN4EA requires random sampling of the path around the entity. Due to these reasons, they can not solve the problem studied in this paper fundamentally.

The methods in the third group are mainly based on graph neural networks. GCN is a structured-only variant of GCN-Align, which only captures the topological connections in KG using GCNs [16]. Without considering relation types of edges, this method has limitations to achieve satisfactory results on both benchmarks. Moreover, the experimental results also show that capturing the correlation of relations can not bring competitive performance for entity alignment (e.g., GCN-Align, HMAN). MuGNN and KECG use GATs [17] to identify the contribution of neighbors, so as to capture relational associations by using GATs and TransE respectively. They achieve better performance than the above methods on the dense datasets. HMAN exploits multi-aspects information (please refer to Fig. 1) in KG. Although these benchmarks do not contain entity description, it still achieves superior performance than previous ones.

Compare to the results in Table III, the performance of aforementioned methods degrades on Table IV, which shows the results of the real-world datasets with sparse relations between entities. For example, the performance of MuGNN has a cliff-like decline. The main reason is that cross-KG attention in MuGNN relies on the similarity between the relations across two KGs to compute the weight of GATs. This mechanism would be heavily affected when the relations between two sparse KGs have high heterogeneity.

The fourth group of methods take into account the literals, and further improve their overall performance. We notice that CEA and BERT-INT have better performance than RDGCN and HGCN on DBP15K and SRPRS (Tables III and IV). Specifically, CEA use existing techniques (GCNs, fastText [39]/MUSE [40], and Levenshtein distance [41]) to get structural, semantic, and string information from graph structures and entity names, which yields inferior performance on Hits@K (K=1, 10) and MRR comparing to BERT-INT and ours on most datasets (except DBP-WD in Table IV). Further, it uses a classical stable matching algorithm [42] to obtain the best matching entity, but it only works for finding 1-1 matching. As a result, CEA can only get Hits@1 score. In fact, the stable matching algorithm can be applied to all embedding methods to boost the performance of 1-1 alignment. For instance, we improve Hits@1 on JA-EN (in DBP15K) from 84.8% to 89.8% when applying the stable matching algorithm, which outperforms CEA (86.3%) by 3.5%. For BERT-INT, it has a strong dependency on entity name (summarized in Table II). Since FR-EN (in DBP15K) and all datasets in SRPRS include well-aligned entity names (which are extracted from Wikipedia page and literally similar), it works well on these datasets as expected. As a comparison, although our method dose not have a strong dependency on entity name, we still achieve comparable performance with BERT-INT.

Table V shows the experimental results on the two challenging datasets, D_W_15K_V1 and D_W_100K_V1. BERT-INT, which relies on entity names, does not even work. This is

TABLE V
EXPERIMENTAL RESULTS ON OPENEA BENCHMARK

Method	D_W_15K_V1			D_W_100K_V1		
	H@1	H@10	MRR	H@1	H@10	MRR
CEA (Emb) [38]	14.9	42.9	0.24	25.1	50.9	0.34
CEA [38]	19.0			44.5		
BERT-INT [34]	0.6	0.6	0.01	0.0	0.1	0.00
SDEA	**65.1**	**77.2**	**0.69**	**57.1**	**64.5**	**0.60**
SDEA w/o rel.	58.2	68.1	0.62	52.0	60.2	0.55

TABLE VI
EVALUATION OF THE EXECUTION TIME.

Method	DBP15K	SRPRS	D_W_15K_V1	D_W_100K_V1
HMAN	52min	45min	45min	–
RDGCN	2h06min	7min	10min	–
HGCN	1h38min	1h27min	1h42min	–
BERT-INT	1h04min	1h01min	1h02min	8h53min
CEA	4min	2min	14min	56min
SDEA	1h13min	1h09min	54min	10h57min

because the two KGs on the datasets do not contain literally matched entity names (name vs. Wikidata ID). Moreover, SDEA outperforms CEA with a large margin, e.g., 46.1% and 12.6% of Hits@1 respectively, which is mainly attributed to the design of identifying various contributions of neighbors and handling alignment of long-tail entities.

Error analysis. We conduct a detailed analysis on the result of challenging datasets. Now, we take D_W_15K_V1 as an example to illustrate our findings.

First, the dataset is very challenging, i.e., many entities have very limited information. From our observation, KGs in D_W_15K_V1 are very sparse in relations. According to our statistics, 99.6% of the to-be-aligned entities in the test set have no matching neighbors. Further, since entity names in W are Wikidata IDs, they do not provide useful information for the alignment. It is hard to align entities with such limited information.

Second, the transformer-based pre-trained language model, i.e., BERT, which we used to get initial embeddings of entities, is insufficient to handle numeric values (as discussed in Section III-A). According to our statistics, about 40% of attribute values in this dataset are numerical attributes, including 9% identifiers, 23% integers and floats, and 8% dates. We will further explore this problem in the future.

2) Evaluation of Efficiency and Scalability: We report the execution time of some well-performed methods in Table VI. Since the source code of baselines requires some old-version libraries, the evaluation of execution time is conducted on a server with an Xeon W-2275 3.3GHz CPU, 64GB memory, and an RTX 5000 GPU with 16GB memory. We failed to obtain the execution time of HMAN, RDGCN, and HGCN due to the limitation of the GPU memory. Among them, CEA is the most efficient method, because it uses simple embeddings, e.g., fastText, which can be learned efficiently. However, CEA has a limitation that it cannot work well without well-matched entity names (in Table V). RDGCN achieves similar efficiency on sparse datasets (SRPRS and D_W_15K_V1), but is very sensitive to the sparsity of data. SDEA has comparable execution time with HMAN, HGCN, and BERT-INT. We tabke about 1 hour and 10 hours on the small (DBP15K, SRPRS, and D_W_15K_V1) and large (D_W_100K_V1) datasets respectively. As training takes most of execution time and is offline in EA, we believe the execution time is acceptable in practice.

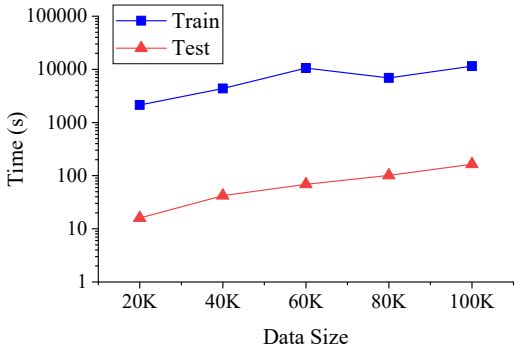


Fig. 6. Evaluation of the scalability.

TABLE VII
PROPORTION OF ENTITY DEGREES WITHIN THE RANGES

Dataset		1∼3	1∼5	1∼10
DBP15K	ZH-EN	30.0%	46.9%	78.5%
	JA-EN	28.8%	44.0%	76.8%
	FR-EN	23.1%	33.4%	63.6%
SRPRS	EN-FR	69.9%	81.5%	92.5%
	EN-DE	65.4%	81.6%	94.7%
	DBP-WD	65.7%	78.9%	90.8%
	DBP-YG	69.8%	82.0%	94.7%
OpenEA	D_W_15K_V1	52.8%	73.7%	91.2%
	D_W_100K_V1	54.7%	74.1%	91.4%

We also evaluate the scalability of our method. We run this experiment with a higher performance machine with two Xeon Gold 5218 2.3GHz CPU, 128GB memory, and an RTX 3090 GPU with 24GB memory. The experiment is conducted on D_W_100K_V1, where we use five proportions of the training set, the validation set, and the test set from 20% to 100%. Fig. 6 shows that the execution time of our method grows almost linearly with the increase of the amount of data, which indicates that our method can scale well on large-scale datasets.

3) Evaluation on Handling Long-tail Entities: This section provides an in-depth analysis on the performance of our method in handling long-tail entities. Table VII presents the proportion of entities with degrees in ranges from 1∼3, 1∼5, and 1∼10. We can see that the datasets of SRPRS and OpenEA contain more than 50% of entities with degrees less than or equal to 3, and less than 10% of entities with degrees greater than 10. In contrast, the datasets of DBP15K contain less than 30% of entities with degrees less than or equal to 3, and more than 20% entities with degrees greater than 10. This clearly show that the datasets in SRPRS and OpenEA have much more long-tail entities than those in DBP15K. As shown in the

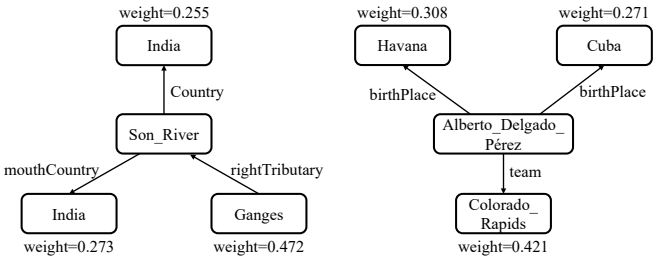


Fig. 7. A case study of two examples on the EN-DE dataset in SRPRS.

results on DBP15K and SRPRS (in Table III and Table IV), the first three groups of baseline methods achieve inferior performance on the datasets in SRPRS. This result implies that the methods taking graph as main features have limitations to handle the alignment of long-tail entities. Compared with these methods, SDEA shows its ability to deal with long-tail entities. It achieves more than 96% Hits@1 score on all datasets of SRPRS and significantly outperforms the best results of the first three groups with 51.3% improvement on average. Further, we show the generality of SDEA to handle long-tail entities in D_W_15K_V1 and D_W_100K_V1, which contain no well-matched entity names (see Table V).

4) Ablation Study: This section presents an ablation study to analyze the effectiveness of attribute embedding and relation embedding modules. The results are reported in the last two rows of Tables III, IV, and V, where SDEA is the full version of our method containing both modules and SDEA w/o rel. eliminates the relation embedding. The results clearly show the effectiveness of relation embedding, which identifies the contribution of neighbors and captures the implicit associations between entities. It is also worth noting that without relation embedding, the results of SDEA w/o rel. still achieve comparable performance on DBP15K and SRPRS, and significantly outperforms baselines on D_W_15K_V1 and D_W_100K_V1. This demonstrates the importance of explicit semantic associations in entity alignment. After adding the relational embedding, the performance of SDEA is further boosted on almost all datasets. One exception is the DBP-YG dataset in DBP15K, where SDEA w/o rel. has already achieved a very high Hit@1 score of 99%.

5) Case Study: In this section, we present a case study to illustrate the advantage of our relation embedding module to distinguish the importance of different neighbors. We derive the weights learned by the attention mechanism in our relation embedding module for each neighbor and show two examples on the EN-DE dataset in SRPRS in Fig. 7.

The first example is about ⟨Son_River⟩ (in the left part of Fig. 7) which is a perennial river in central India. From its three neighbors, we can learn that ⟨Son_River⟩ is a tributary of ⟨Ganges⟩ and it is located in ⟨India⟩. The river mouth of ⟨Son_River⟩ is also located in ⟨India⟩. On the one hand, among the three relationships, the one that it is a tributary of ⟨Ganges⟩ is the most fine-grained and more effective to represent its role in the knowledge graph. Therefore, this relationship is assigned with the highest weight of 0.472. On the other hand, ⟨Son_River⟩'s and its mouth's location country is relatively coarse-grained, and thus they are assigned with lower weights.

The second example is about ⟨Alberto_Delgado_Pérez⟩ (in the right part of Fig. 7), who is a football player and also has three neighbors. From the graph, we can learn that he is a member of ⟨Colorado_Rapids⟩ and was born in ⟨Havana⟩, ⟨Cuba⟩. Intuitively, the team information is the most useful for identifying players. We can see that the weight assignment is consistent with the intuition. Besides, the city of birth is more fine-grained than the country of birth, and thus the former is assigned with a relatively higher weight.

VI. Related Work

Entity alignment has been extensively studied in the recent decades, and the existing studies can be broadly divided into the following categories: rule-based approaches [43], crowdsourcing-based approaches [44], [45], machine learning (ML) approaches [46], and deep learning (DL) approaches [15], [23], [34]. Among these approaches, not surprisingly, DL-based approaches have achieved the state-of-the-art performance. Thus, we mainly focus on summarizing the DL approaches in this section.

Most DL-based approaches for entity alignment leverage embedding-based techniques. Some early studies [9], [10] use the original TransE [47] to train KG embeddings, which may not perform well when capturing 1-N, N-1 and N-N relations between entities. Thus, some approaches propose to consider the structure of KGs. GCN-Align [18] employs GCN-based [16] techniques. The basic idea is to generate entity embeddings based on topological structure of entities in KGs by utilizing graph neural networks. RSN4EA [23] studies the problem of handling the alignment of long-tail entities and proposes a path-based embedding approach to address the long-distance transmission of alignment information. Moreover, some researchers consider different contributions of neighbors to entities to be aligned, and devise attention-based mechanisms on TransE-based model [11] or with GATs [21], [22].

Some recent studies further consider textual information of entities to improve the performance, e.g., exploiting semantics from entity names or descriptions. To this end, they embed names/descriptions separately from structural embeddings [15], [34], [38] or use entity names/descriptions as initial input of GCNs [19], [20], [25]. BERT-INT [34] explores to use attribute values, which provide rich semantic information that are not included in graph structure. It encodes attribute values by BERT and calculates the pairwise attribute similarities, which only captures the direct semantic association between attribute values. CEA [38] focuses more on the alignment algorithm and performs a classical stable matching algorithm [42] to achieve better alignment. However, this method works only for 1-1 matching.

Compared with these approaches, our work aims to fully exploit the inherent semantics of entities from all attribute values.

We use the conceptual granularity and the semantic relevance of neighbors to identify the contribution, which is more reliable than those solely relying on graph structures. Further, we take the advantage of direct (between attributes) and indirect (between attributes and neighbors) semantic associations to build bridges between entities, which can effectively solve the problem of aligning long-tail entities.

Nowadays, some new directions have emerged in the research of entity alignment, which motivates our future work. Regarding the practical issues present in dealing with real-world knowledge bases, researchers propose completely unsupervised solutions [48]–[50], dedicate to improving the speed [51], and propose dynamic entity alignment solutions [52]. Also, some studies jointly learn entity and relation embeddings [53]–[55], and incorporate new types of information in KGs, such as images [56].

VII. Conclusion

In this paper, we discuss two design considerations motivated by a careful analysis on real-word KGs. We propose SDEA – a semantics-driven entity embedding method for effective entity alignment, which identifies the contribution of neighbors, handling the alignment of long-tail entities by joint learning entity representation from three aspects: the semantic relevance of neighbors, the direct associations of attributes, and the indirect associations of attributes and neighbors. We conduct extensive experiments to evaluate our approach. The experimental results show the superior performance of SDEA on widely-used benchmark datasets. We also conduct an ablation study and a case study to analyze each component of SDEA, which validates the rationality and effectiveness of our design.

References

[1] W. Yih, M. Chang, X. He, and J. Gao, "Semantic parsing via staged query graph generation: Question answering with knowledge base," in *ACL*. The Association for Computer Linguistics, 2015, pp. 1321–1331.

[2] N. Abdullah and R. Ibrahim, "Knowledge retrieval in lexical ontology-based semantic web search engine," in *ICUIMC*. ACM, 2013, p. 8.

[3] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W. Ma, "Collaborative knowledge base embedding for recommender systems," in *KDD*. ACM, 2016, pp. 353–362.

[4] 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 *International Semantic Web Conference*, ser. Lecture Notes in Computer Science, vol. 9982, 2016, pp. 177–185.

[5] 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*, vol. 6, no. 2, pp. 167–195, 2015.

[6] K. D. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge," in *SIGMOD Conference*. ACM, 2008, pp. 1247–1250.

[7] "Imdb." [Online]. Available: http://www.imdb.com

[8] 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," *Proc. VLDB Endow.*, vol. 13, no. 11, pp. 2326–2340, 2020.

[9] M. Chen, Y. Tian, M. Yang, and C. Zaniolo, "Multilingual knowledge graph embeddings for cross-lingual knowledge alignment," in *IJCAI*. ijcai.org, 2017, pp. 1511–1517.

[10] Z. Sun, W. Hu, and C. Li, "Cross-lingual entity alignment via joint attribute-preserving embedding," in *International Semantic Web Conference*, ser. Lecture Notes in Computer Science, vol. 10587. Springer, 2017, pp. 628–644.

[11] Q. Zhu, X. Zhou, J. Wu, J. Tan, and L. Guo, "Neighborhood-aware attentional representation for multilingual knowledge graphs," in *IJCAI*. ijcai.org, 2019, pp. 1943–1949.

[12] Z. Sun, W. Hu, Q. Zhang, and Y. Qu, "Bootstrapping entity alignment with knowledge graph embedding," in *IJCAI*. ijcai.org, 2018, pp. 4396–4402.

[13] Z. Sun, J. Huang, W. Hu, M. Chen, L. Guo, and Y. Qu, "Transedge: Translating relation-contextualized embeddings for knowledge graphs," in *ISWC*, ser. Lecture Notes in Computer Science, vol. 11778. Springer, 2019, pp. 612–629.

[14] H. Zhu, R. Xie, Z. Liu, and M. Sun, "Iterative entity alignment via joint knowledge embeddings," in *IJCAI*. ijcai.org, 2017, pp. 4258–4264.

[15] H. Yang, Y. Zou, P. Shi, W. Lu, J. Lin, and X. Sun, "Aligning cross-lingual entities with multi-aspect information," in *EMNLP/IJCNLP*. Association for Computational Linguistics, 2019, pp. 4430–4440.

[16] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR (Poster)*. OpenReview.net, 2017.

[17] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *ICLR (Poster)*. OpenReview.net, 2018.

[18] Z. Wang, Q. Lv, X. Lan, and Y. Zhang, "Cross-lingual knowledge graph alignment via graph convolutional networks," in *EMNLP*. Association for Computational Linguistics, 2018, pp. 349–357.

[19] Y. Wu, X. Liu, Y. Feng, Z. Wang, R. Yan, and D. Zhao, "Relation-aware entity alignment for heterogeneous knowledge graphs," in *IJCAI*. ijcai.org, 2019, pp. 5278–5284.

[20] Y. Wu, X. Liu, Y. Feng, Z. Wang, and D. Zhao, "Jointly learning entity and relation representations for entity alignment," in *EMNLP/IJCNLP*. Association for Computational Linguistics, 2019, pp. 240–249.

[21] Y. Cao, Z. Liu, C. Li, Z. Liu, J. Li, and T. Chua, "Multi-channel graph neural network for entity alignment," in *ACL*. Association for Computational Linguistics, 2019, pp. 1452–1461.

[22] C. Li, Y. Cao, L. Hou, J. Shi, J. Li, and T. Chua, "Semi-supervised entity alignment via joint knowledge embedding model and cross-graph model," in *EMNLP/IJCNLP*. Association for Computational Linguistics, 2019, pp. 2723–2732.

[23] L. Guo, Z. Sun, and W. Hu, "Learning to exploit long-term relational dependencies in knowledge graphs," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 97. PMLR, 2019, pp. 2505–2514.

[24] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *ICLR (Workshop Poster)*, 2013.

[25] 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*. Association for Computational Linguistics, 2019, pp. 3156–3161.

[26] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT*. Association for Computational Linguistics, 2019, pp. 4171–4186.

[27] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*, 2017, pp. 5998–6008.

[28] E. Wallace, Y. Wang, S. Li, S. Singh, and M. Gardner, "Do NLP models know numbers? probing numeracy in embeddings," in *EMNLP/IJCNLP (1)*. Association for Computational Linguistics, 2019, pp. 5306–5314.

[29] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *NAACL-HLT*. Association for Computational Linguistics, 2018, pp. 2227–2237.

[30] W. Tai, H. T. Kung, X. Dong, M. Z. Comiter, and C. Kuo, "exbert: Extending pre-trained models with domain-specific vocabulary under constrained training resources," in *EMNLP (Findings)*, ser. Findings of ACL, vol. EMNLP 2020. Association for Computational Linguistics, 2020, pp. 1433–1439.

[31] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang, "Biobert: a pre-trained biomedical language representation model for biomedical text mining," *Bioinform.*, vol. 36, no. 4, pp. 1234–1240, 2020.

[32] K. Cho, A. C. Courville, and Y. Bengio, "Describing multimedia content using attention-based encoder-decoder networks," *IEEE Trans. Multim.*, vol. 17, no. 11, pp. 1875–1886, 2015.

[33] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," pp. 1724–1734, 2014.

[34] 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*. ijcai.org, 2020, pp. 3174–3180.

[35] 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 & Data Engineering*, 2020.

[36] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *NeurIPS*, 2019, pp. 8024–8035.

[37] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Transformers: State-of-the-art natural language processing," in *EMNLP (Demos)*. Association for Computational Linguistics, 2020, pp. 38–45.

[38] W. Zeng, X. Zhao, J. Tang, and X. Lin, "Collective entity alignment via adaptive features," in *ICDE*. IEEE, 2020, pp. 1870–1873.

[39] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Trans. Assoc. Comput. Linguistics*, vol. 5, pp. 135–146, 2017.

[40] G. Lample, A. Conneau, M. Ranzato, L. Denoyer, and H. Jégou, "Word translation without parallel data," in *ICLR (Poster)*. OpenReview.net, 2018.

[41] V. I. Levenshtein *et al.*, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet physics doklady*, vol. 10, no. 8. Soviet Union, 1966, pp. 707–710.

[42] D. Gale and L. S. Shapley, "College admissions and the stability of marriage," *The American Mathematical Monthly*, vol. 69, no. 1, pp. 9–15, 1962.

[43] R. Singh, V. V. Meduri, A. K. Elmagarmid, S. Madden, P. Papotti, J. Quiané-Ruiz, A. Solar-Lezama, and N. Tang, "Synthesizing entity matching rules by examples," *Proc. VLDB Endow.*, vol. 11, no. 2, pp. 189–202, 2017.

[44] R. Meng, L. Chen, Y. Tong, and C. J. Zhang, "Knowledge base semantic integration using crowdsourcing," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 5, pp. 1087–1100, 2017.

[45] Y. Zhuang, G. Li, Z. Zhong, and J. Feng, "Hike: A hybrid human-machine method for entity alignment in large-scale knowledge bases," in *CIKM*. ACM, 2017, pp. 1917–1926.

[46] P. Konda, S. Das, P. S. G. C., A. Doan, A. Ardalan, J. R. Ballard, H. Li, F. Panahi, H. Zhang, J. F. Naughton, S. Prasad, G. Krishnan, R. Deep, and V. Raghavendra, "Magellan: Toward building entity matching management systems," *Proc. VLDB Endow.*, vol. 9, no. 12, pp. 1197–1208, 2016.

[47] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *NIPS*, 2013, pp. 2787–2795.

[48] Z. Qi, Z. Zhang, J. Chen, X. Chen, Y. Xiang, N. Zhang, and Y. Zheng, "Unsupervised knowledge graph alignment by probabilistic reasoning and semantic embedding," in *IJCAI*. ijcai.org, 2021, pp. 2019–2025.

[49] W. Zeng, X. Zhao, J. Tang, and C. Fan, "Reinforced active entity alignment," in *CIKM*. ACM, 2021, pp. 2477–2486.

[50] X. Mao, W. Wang, Y. Wu, and M. Lan, "From alignment to assignment: Frustratingly simple unsupervised entity alignment," in *EMNLP*. Association for Computational Linguistics, 2021, pp. 2843–2853.

[51] ——, "Boosting the speed of entity alignment 10 ×: Dual attention matching network with normalized hard sample mining," in *WWW*. ACM / IW3C2, 2021, pp. 821–832.

[52] Y. Yan, L. Liu, Y. Ban, B. Jing, and H. Tong, "Dynamic knowledge graph alignment," in *AAAI*. AAAI Press, 2021, pp. 4564–4572.

[53] J. Yang, D. Wang, W. Zhou, W. Qian, X. Wang, J. Han, and S. Hu, "Entity and relation matching consensus for entity alignment," in *CIKM*. ACM, 2021, pp. 2331–2341.

[54] X. Mao, W. Wang, Y. Wu, and M. Lan, "Are negative samples necessary in entity alignment?: An approach with high performance, scalability and robustness," in *CIKM*. ACM, 2021, pp. 1263–1273.

[55] Y. Zhu, H. Liu, Z. Wu, and Y. Du, "Relation-aware neighborhood matching model for entity alignment," in *AAAI*. AAAI Press, 2021, pp. 4749–4756.

[56] F. Liu, M. Chen, D. Roth, and N. Collier, "Visual pivoting for (unsupervised) entity alignment," in *AAAI*. AAAI Press, 2021, pp. 4257–4266.