Entity Alignment between Knowledge Graphs Using Attribute Embeddings

Bayu Distiawan Trsedya and **Jianzhong Qi** and **Rui Zhang***
School of Computing and Information Systems, The University of Melbourne
{btrisedya@student., jianzhong.qi@, rui.zhang@}unimelb.edu.au

Abstract

The task of entity alignment between knowledge graphs aims to find entities in two knowledge graphs that represent the same real-world entity. Recently, embedding-based models are proposed for this task. Such models are built on top of a knowledge graph embedding model that learns entity embeddings to capture the semantic similarity between entities in the *same* knowledge graph. We propose to learn embeddings that can capture the similarity between entities in *different* knowledge graphs. Our proposed model helps align entities from different knowledge graphs, and hence enables the integration of multiple knowledge graphs. Our model exploits large numbers of attribute triples existing in the knowledge graphs and generates *attribute character embeddings*. The attribute character embedding shifts the entity embeddings from two knowledge graphs into the same space by computing the similarity between entities based on their attributes. We use a transitivity rule to further enrich the number of attributes of an entity to enhance the attribute character embedding. Experiments using real-world knowledge bases show that our proposed model achieves consistent improvements over the baseline models by over 50% in terms of $hits@1$ on the entity alignment task.

1 Introduction

Knowledge bases in the form of *knowledge graphs* (KGs) have been used in many applications including question answering systems (Yih et al. 2015), recommender systems (Zhang et al. 2016), and sentence generation (Trisedya et al. 2018). Many KGs have been created separately for particular purposes. The same entity may exist in different forms in different KGs. For example, `lgd:240111203` in a KG named LinkedGeoData (Stadler et al. 2012) and `dbp:Kromsdorf` in another KG named DBpedia (Lehmann et al. 2015) both refer to a city named Kromsdorf in Germany. Typically, these KGs are complementary to each other in terms of completeness. We may integrate such KGs to form a larger KG for knowledge inferences.

To integrate KGs, a basic problem is to identify the entities in different KGs that denote the same real-world entity, which is commonly referred to as the entity alignment problem. In this paper, we study entity alignment between two KGs. Specifically, we consider KGs where real-world facts are stored in the form of RDF triples. An RDF triple consists of three elements in the form of $\langle$*subject*, *relationship/predicate*, *object*$\rangle$, where *subject* denotes an entity, and *object* denotes either an entity or a literal. Here, if *object* is an entity, we call the triple a *relationship triple*; if *object* is a literal, we call the triple an *attribute triple*. Table 1 gives an example of two subsets of RDF triples from two KGs G_1 and G_2 (we use prefixes `"lgd:"` and `"dbp:"` to simplify the original URI). The subjects in these two subsets refer to the same entity Kromsdorf, even though they are in different forms `"lgd:240111203"` and `"dbp:Kromsdorf"`. We aim to identify such entities and give them a unified ID such that both KGs can be merged together through them. In Table 1, $G_{1,2}$ denotes the merged KG, where `"lgd:240111203"` is used as the unified ID for the entity Kromsdorf which has a set of attributes

Table 1: Knowledge Graph Alignment Example

G_1
⟨lgd:240111203,geo:long,11.3700843⟩
⟨lgd:240111203,lgd:population,1595⟩
⟨lgd:240111203,rdfs:label,"Kromsdorf"⟩
⟨lgd:240111203,geo:lat,50.9988888889⟩
⟨lgd:240111203,lgd:alderman,"B. Grobe"⟩
⟨lgd:240111203,lgd:country,lgd:51477⟩
...
G_2
⟨dbp:Kromsdorf,geo:long,11.3701⟩
⟨dbp:Kromsdorf,rdfs:label,"Kromsdorf"⟩
⟨dbp:Kromsdorf,geo:lat,50.9989⟩
⟨dbp:Kromsdorf,dbp:populationTotal,1595⟩
⟨dbp:Kromsdorf,dbp:country,dbp:Germany⟩
⟨dbp:Kromsdorf,dbp:district,dbp:Weimarer⟩
...
Merged $G_{1,2}$
⟨lgd:240111203,geo:long,11.3700843⟩
⟨lgd:240111203,:population,1595⟩
⟨lgd:240111203,rdfs:label,"Kromsdorf"⟩
⟨lgd:240111203,geo:lat,50.9988888889⟩
⟨lgd:240111203,lgd:alderman,"B. Grobe"⟩
⟨lgd:240111203,:country,lgd:51477⟩
⟨lgd:240111203,dbp:district,dbp:Weimarer⟩
...

*Corresponding author

that is the union of the sets of attributes from both KGs.

Early studies on entity alignment are based on the similarity between attributes of entities (Ngomo and Auer 2011; Pershina, Yakout, and Chakrabarti 2015; Volz et al. 2009). These methods rely on user-defined rules to determine the attributes to be compared between the entities. For example, the attributes to be compared between entities of the two KGs in Table 1 are `rdfs:label`, `geo:lat`, and `geo:long`. Such approaches are error-prone because different pairs of entities may need different attributes to be compared, e.g., for two celebrity entities, they may not have attributes such as `geo:lat` and `geo:long`.

Recently, embedding-based models are proposed for the entity alignment task. Such models are built on top of a KG embedding model, such as TransE (Bordes et al. 2013), that learns *entity embeddings* that capture the semantic similarity between entities in a knowledge graph based on the relationship triples in a KG. To adapt the KG embedding for entity alignment between two KGs, the embedding-based models require the embeddings of two KGs to fall in the same vector space. To address this problem, Chen et al. (2017a; 2017b) and Zhu et al. (2017) learn an embedding space for each KG separately and propose to use a transition matrix to map the embedding space from one KG to the other. Their models rely on large numbers of seed alignments (i.e., a seed set of aligned triples from two KGs) to compute the transition matrix. However, the seed alignments between two KGs are rarely available, and hence are difficult to obtain due to expensive human efforts required.

In this paper, we address the above limitations by proposing a novel embedding model that first generates *attribute embeddings* from the attribute triples and then use this attribute embeddings to shift the entity embeddings of two KGs to the same vector space. We observe that many KGs contain large numbers of attribute triples, which have not been explored for entity alignment so far. For example, DBpedia, YAGO, LinkedGeoData, and Geonames contain 47.62%, 62.78%, 94.66%, and 76.78% of attribute triples, respectively. The attribute similarity between two KGs helps the attribute embedding to yield a unified embedding space for two KGs. This enables us to use the attribute embeddings to shift the entity embeddings of two KGs into the same vector space and hence enables the entity embeddings to capture the similarity between entities from two KGs. Our model includes predicate alignment by renaming the predicates of two KGs into a unified naming scheme to ensure that the relationship embeddings of two KGs are also in the same vector space. We further use the transitive rule (e.g., by knowing that `Emporium Tower` is located in `London` and `London` is located in `England`, we also know that `Emporium Tower` is located in `England`.) to enrich attribute triples for attribute embedding computation.

Our contributions are summarized as follows:

- We propose a framework for entity alignment between two KGs that consists of a predicate alignment module, an embedding learning module, and an entity alignment module.
- We propose a novel embedding model that integrates entity embeddings with attribute embeddings to learn a unified embedding space for two KGs.
- We evaluate the proposed model over three real KG pairs. The results show that our model outperforms the state-of-the-art models consistently on the entity alignment task by over 50% in terms of $hits@1$.

The rest of this paper is organized as follows. Section 2 summarizes previous studies on entity alignment. Section 3 defines the studied problem. Section 4 details the proposed model. Section 5 presents the experimental results. Section 6 concludes the paper.

2 Related Work

We discuss two groups of commonly used entity alignment approaches: string-similarity-based approaches (Section 2.1) and embedding-based approaches (Section 2.2).

2.1 String-Similarity-based Entity Alignment

Earlier entity alignment approaches use string similarity as the main alignment tool. For example, LIMES (Ngomo and Auer 2011) uses the *triangle inequality* to compute an approximation of entity similarity. Then, the actual similarity of the entity pairs that have a high approximated similarity is computed, and the entity pair with the highest actual string similarity is returned. RDF-AI (Scharffe, Yanbin, and Zhou 2009) implements an alignment framework that consists of preprocessing, matching, fusion, interlink, and post-processing modules, among which the matching module uses fuzzy string matching based on sequence alignment (Rivas and Eddy 1999), word relation (Fellbaum 1998), and taxonomical similarity algorithms. SILK (Volz et al., 2009) allows users to specify the mapping rules using the *Silk - Link Specification Language* (Silk-LSL). SILK provides similarity metrics including string equality and similarity, numeric similarity, date similarity, and URI equality.

There are also studies using graph similarity to improve the entity alignment performance. LD-Mapper (Raimond, Sutton, and Sandler 2008) combines string similarity with entity nearest neighbor similarity. RuleMiner (Niu et al. 2012) uses an Expectation-Maximization (EM) algorithm to refine a set of manually defined entity matching-rules. HolisticEM (Pershina, Yakout, and Chakrabarti 2015) constructs a graph of potential entity pairs based on the overlapping attributes and the neighboring entities of the entities. Then, the local and global properties from the graph are propagated using *Personalized Page Rank* to compute the actual similarity of entity pairs.

2.2 Embedding-based Entity Alignment

KG embedding models have been used to address KG completion tasks that aim to predict missing entities or relations based on existing triples in a KG. Among the existing approaches, the translation based models, e.g., TransE (Bordes et al. 2013), achieve the state-of-the-art performance. TransE represents a relationship between a pair of entities as a translation between the embeddings of the entities. In recent years, several studies propose improvements over TransE: TransH (Wang et al. 2014), TransR (Lin et al. 2015),

and TransD (Ji et al. 2015) use a distributed representation to separate the relationship vector space from the entity vector space; DKRL (Xie et al. 2016) and TEKE (Wang and Li 2016) use additional information (e.g., entity description and word co-occurrence) along with the relationship triples to compute entity embeddings. There are also non-translation-based approaches to learn entity embeddings. The Unstructured model (Bordes et al. 2011) does not explicitly represent the relationship embeddings. RESCAL (Nickel, Tresp, and Kriegel 2012) and HolE (Nickel, Rosasco, and Poggio 2016) use tensor-based factorization and represent relationships with matrices. NTN (Socher et al. 2013) uses a bilinear tensor operator to represent each relationship and jointly models head and tail entity embeddings.

The embedding models above aim to preserve the structural information of the entities, i.e., entities that share similar neighbor structures in the KG should have a close representation in the embedding space. The advancement of such embedding models motivates researchers to study embedding-based entity alignment. Chen et al. (2017a; 2017b) propose an embedding-based model for multilingual entity alignment based on TransE. First, the embeddings are computed for each KG. Then, a transition matrix is computed using a set of manually collected seed pairs of aligned entities to provide transitions for each embedding to its cross-lingual counterparts. Zhu et al. (2017) propose an iterative method for entity alignment via joint embeddings. Similar to the previous model, initially, the joint embeddings between KGs is computed using the seed pairs of aligned entities. Then, the joint embeddings are updated iteratively using the newly aligned entities. Sun, Hu, and Li (2017) propose a joint attribute-preserving embedding model for cross-lingual entity alignment. The seed alignments are used to jointly embed the entity embeddings of two KGs into a unified vector space. Then, the embeddings are updated using the attribute correlations that are computed based on the attribute type similarity. Although this work also considers the entity attribute information, it differs from ours in that we use the actual attribute value instead of the attribute type.

3 Preliminary

We start with the problem definition. A KG G consists of a combination of relationship triples in the form of $\langle h, r, t\rangle$, where r is a relationship (*predicate*) between two entities h (*subject*) and t (*object*), and attribute triples in the form of $\langle h, r, a\rangle$ where a is an attribute value of entity h with respect to the predicate (relationship) r. Given two KGs G_1 and G_2, the task of entity alignment aims to find every pair $\langle h_1, h_2\rangle$ where $h_1 \in G_1$, $h_2 \in G_2$, and h_1 and h_2 represent the same real-world entity. We use an embedding-based model that assigns a continuous representation for each element of a triple in the forms of $\langle \mathbf{h}, \mathbf{r}, \mathbf{t}\rangle$ and $\langle \mathbf{h}, \mathbf{r}, \mathbf{a}\rangle$, where the bold-face letters denote the vector representations of the corresponding element.

Most embedding-based models are built on top of TransE (Chen et al. 2017a; 2017b; Sun, Hu, and Li 2017; Zhu et al. 2017). We first discuss TransE (Bordes et al. 2013) and its limitations when being used for entity alignment before presenting our proposed model.

3.1 TransE

Given a relationship triple $\langle h, r, t\rangle$, TransE suggests that the embedding of the tail entity t should be close to the embedding of the head entity h plus the embedding of the relationship r, i.e., $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$. Such an embedding model aims to preserve the structural information of the entities, i.e., entities that share similar neighbor structures in the KG should have a close representation in the embedding space. We call it a *structure embedding*. To learn the structure embedding, TransE minimizes a margin-based objective function J_{SE}:

$$J_{SE} = \sum_{t_r \in T_r} \sum_{t'_r \in T'_r} \max\left(0, [\gamma + f(t_r) - f(t'_r)]\right) \quad (1)$$

$$T_r = \{\langle h, r, t\rangle | \langle h, r, t\rangle \in G\}; f(t_r) = \|\mathbf{h} + \mathbf{r} - \mathbf{t}\| \quad (2)$$

Here, $\|\mathbf{x}\|$ is the L1-Norm of vector $\mathbf{x}$, γ is a margin hyper-parameter, and T_r is the set of valid relationship triples from the training dataset, and T'_r is the set of corrupted relationship triples (E is the set of entities in G):

$$T_r{}' = \{\langle h', r, t\rangle \mid h' \in E\} \cup \{\langle h, r, t'\rangle \mid t' \in E\} \quad (3)$$

The corrupted triples are used as negative samples, which are created by replacing the head or tail entity of a valid triple in T_r with a random entity.

TransE has been used to address KG completion tasks that aim to predict missing entities or relations based on existing triples in a KG. TransE constructs a low-dimensional and continuous vector (embedding) to describe the latent semantic information (as reflected by the neighboring entities) of a KG. The resulting embeddings capture the semantic similarity between entities in the embedding space. For example, the embedding of `dbp:Germany` should be close to the embedding of `dbp:France` as both entities represent two countries in Europe; they share similar types of neighboring entities (e.g., president, continent, etc.).

The advantages of structure embedding drive further studies of embedding-based entity alignment. However, a straightforward implementation of structure embedding for entity alignment has limitations. In particular, the entity embeddings computed on different KGs may fall in different spaces, where similarity cannot be computed directly. Existing techniques (Chen et al. 2017a; Sun, Hu, and Li 2017; Zhu et al. 2017) address this limitation by computing a transition matrix to map the embedding spaces of different KGs into the same space, as discussed earlier. However, such techniques require manually collecting a seed set of aligned entities from the different KGs to create the transition matrix, which do not scale and are vulnerable to the quality of the selected seed aligned entities.

Next, we detail our proposed model to address these limitations.

4 Proposed Model

We present an overview of our proposed model in Section 4.1. We detail the components of the proposed model afterwards, including predicate alignment in Section 4.2, embedding learning in Section 4.3, entity alignment in Section 4.4, and triple enrichment in Section 4.5.

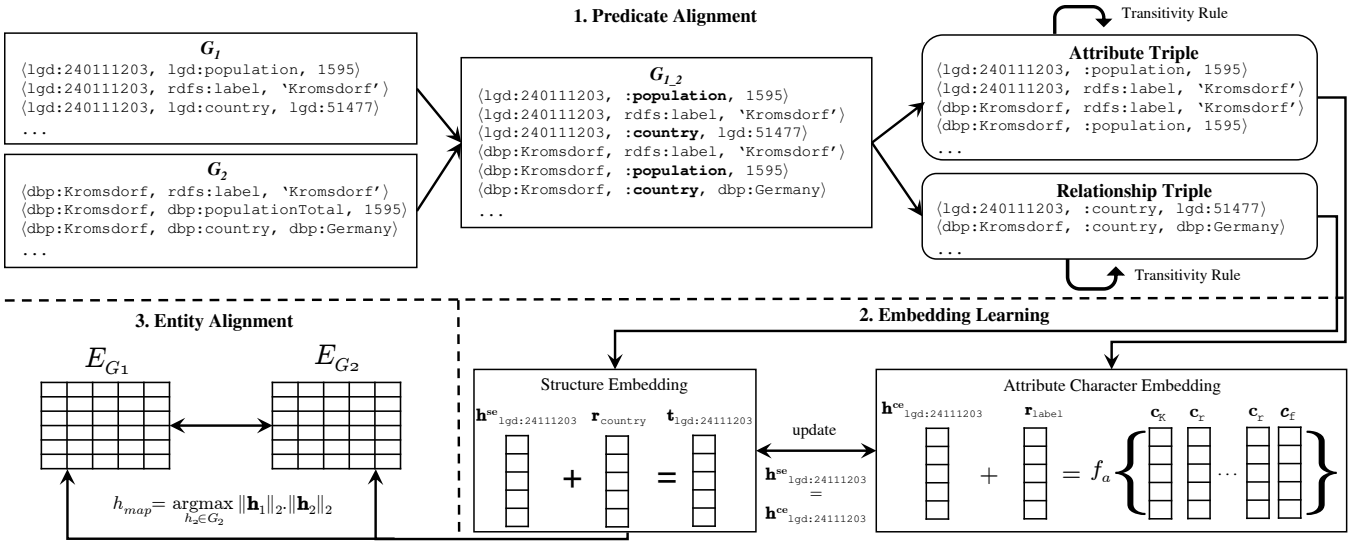

Figure 1: Overview of our proposed solution

4.1 Solution Overview

Our solution framework uses an embedding-based model as illustrated in Fig 1. The framework consists of three components including *predicate alignment*, *embedding learning*, and *entity alignment*.

Embedding-based entity alignment requires the embeddings (both relationship and entity embeddings) of two KGs to fall in the same vector space. To have a unified vector space for the relationship embeddings, we merge two KGs based on the predicate similarity (i.e., predicate alignment). The predicate alignment module (detailed in Section 4.2) finds partially similar predicates, e.g., `dbp:bornIn` vs. `yago:wasBornIn` and renames them with a unified naming scheme (e.g., `:bornIn`). Based on this unified naming scheme, we merge G_1 and G_2 into G_{1_2}. Then, the merged graph G_{1_2} is split into a set of relationship triples T_r and a set of attribute triples T_a for embedding learning.

The embedding learning module (detailed in Section 4.3) jointly learns the entity embeddings of two KGs using structure embedding and attribute embedding. The structure embedding is learned using the set of relationship triples T_r, while the attribute embedding is learned using the set of attribute triples T_a. Initially, the structure embeddings of the entities that come from G_1 and G_2 fall into different vector spaces because the entities from both KGs are represented using different naming schemes. In contrary, the attribute embeddings learned from the attribute triples T_a can fall into the same vector space. This is achieved by learning character embeddings from the attribute strings, which can be similar even if the attributes are from different KGs (we call it as *attribute character embedding*). Then, we use the resulting attribute character embedding to shift the structure embeddings of the entities into the same vector space which enables the entity embeddings to capture the similarity between entities from two KGs. As an example, suppose that we have triples ⟨`lgd:240111203, :country, lgd:51477`⟩ and ⟨`lgd: 51477, :label, "Germany"`⟩ from G_1, and ⟨`dbp:Kromsdorf, :country, dbp:Germany`⟩ and ⟨`dbp:Germany, :label, "Germany"`⟩ from G_2. The attribute character embedding allows both entities `lgd:5147` and `dbp:Germany` to have similar vector representations since both entities have a similar attribute value `"Germany"`. Then, the structure embeddings of entities `lgd:240111203` and `dbp:Kromsdorf` will also be similar since the two entities share the same predicate and have two tail entities `lgd:51477` and `dbp:Germany` which have similar vector representations.

Once we have the embeddings for all entities in G_1 and G_2, the entity alignment module (detailed in Section 4.4) finds every pair $\langle h_1, h_2 \rangle$ where $h_1 \in G_1$ and $h_2 \in G_2$ with a similarity score above a threshold β.

To further improve the performance of the model, we use the relationship transitivity rule to enrich the attributes of an entity that helps build a more robust attribute embedding for computing the similarity between entities. This is detailed in Section 4.5.

4.2 Predicate Alignment

The predicate alignment module merges two KGs by renaming the predicates of both KGs with a unified naming scheme to have a unified vector space for the relationship embeddings. In fact, there are naming conventions of predicates, e.g., `rdfs:label`, `geo:wgs84_pos#lat`, and `geo:wgs84_pos#long`. Besides the naming conventions, there are partially matched predicates, e.g., `dbp:diedIn` vs. `yago:diedIn`, and `dbp:bornIn` vs. `yago:wasBornIn`. Our predicate alignment module finds these predicates and renames them with a unified naming scheme (e.g., `:diedIn` and `:bornIn`). To find the partially matched predicates, we compute the edit distance of the last part of the predicate URI (e.g., `bornIn` vs.

wasBornIn) and set 0.95 as the similarity threshold.

4.3 Embedding Learning

Aligning predicates of two KGs allows our model to have a unified relationship vector space and hence enables the joint learning of structure embedding and attribute character embedding that aims to yield a unified entity vector space. We detail the joint learning in the following section.

Structure Embedding We adapt TransE to learn the structure embedding for entity alignment between KGs by focusing the embedding learning more on the aligned triples (i.e., triples with aligned predicates). This is done by adding a weight α to control the embedding learning over the triples. To learn the structure embedding, in our model, we minimize the following objective function J_{SE}:

$$J_{SE} = \sum_{t_r \in T_r} \sum_{t'_r \in T'_r} \max\left(0, \gamma + \alpha\left(f(t_r) - f(t'_r)\right)\right) \quad (4)$$

$$\alpha = \frac{count(r)}{|T|} \quad (5)$$

where T_r is the set of valid relationship triples, T'_r is the set of corrupted relationship triples, $count(r)$ is the number of occurrences of relationship r, and $|T|$ is the total number of triples in the merge KG $G_{1\text{-}2}$. Typically, the number of occurrences of the aligned predicates is higher than the non-aligned predicates since the aligned predicates appear in both KGs, and hence allows the model to learn more from the aligned triples. For example, for the data in Table 1, weight α helps the embedding model to focus more on relationships rdfs:label, geo:lat, and geo:long ($\alpha = 2/12$ for each of these predicates) than on relationships lgd:alderman or dbp:district ($\alpha = 1/12$ for each of these predicates).

Attribute Character Embedding Following TransE, for the attribute character embedding, we interpret predicate r as a translation from the head entity h to the attribute a. However, the same attribute a may appear in different forms in two KGs, e.g., 50.9989 vs. 50.9988888889 as the latitude of an entity; "Barack Obama" vs. "Barack Hussein Obama" as a person name, etc. Hence, we use a compositional function to encode the attribute value and define the relationship of each element in an attribute triple as $\mathbf{h} + \mathbf{r} \approx f_a(a)$. Here, $f_a(a)$ is a compositional function and a is a sequence of the characters of the attribute value $a = \{c_1, c_2, c_3, ..., c_t\}$. The compositional function encodes the attribute value into a single vector and maps similar attribute values to a similar vector representation. We define three compositional functions as follows.

Sum compositional function (**SUM**). The first compositional function is defined as a summation of all character embeddings of the attribute value.

$$f_a(a) = \mathbf{c_1} + \mathbf{c_2} + \mathbf{c_3} + ... + \mathbf{c_t} \quad (6)$$

where $\mathbf{c_1}, \mathbf{c_2}, ..., \mathbf{c_t}$ are the character embeddings of the attribute value. This compositional function is simple but it suffers in that two strings that contain the same set of characters with a different order will have the same vector representation. For example, two coordinates "50.15" and "15.05" will have the same vector representation.

LSTM-based compositional function (**LSTM**). To address the problem of SUM, we propose an LSTM-based compositional function. This function uses LSTM networks to encode a sequence of characters into a single vector. We use the final hidden state of the LSTM networks as a vector representation of the attribute value.

$$f_a(a) = f_{lstm}(\mathbf{c_1}, \mathbf{c_2}, \mathbf{c_3}, ..., \mathbf{c_t}) \quad (7)$$

where f_{lstm} is the LSTM networks as defined by Kim et al. (2016).

N-gram-based compositional function (**N-gram**). We further propose an N-gram-based compositional function as an alternative to address the problem of SUM. Here, we use the summation of n-gram combination of the attribute value.

$$f_a(a) = \sum_{n=1}^{N} \left(\frac{\sum_{i=1}^{t} \sum_{j=i}^{n} \mathbf{c_j}}{t - i - 1} \right) \quad (8)$$

where N indicates the maximum value of n used in the n-gram combinations ($N = 10$ in our experiments) and t is the length of the attribute value.

To learn the attribute character embedding, we minimize the following objective function J_{CE}:

$$J_{CE} = \sum_{t_a \in T_a} \sum_{t'_a \in T'_a} \max\left(0, [\gamma + \alpha\left(f(t_a) - f(t'_a)\right)]\right) \quad (9)$$

$$T_a = \{\langle h, r, a\rangle \in G\}; f(t_a) = \|\mathbf{h} + \mathbf{r} - f_a(a)\| \quad (10)$$

$$T_a{}' = \{\langle h', r, a\rangle | \, h' \in E\} \cup \{\langle h, r, a'\rangle \, | a' \in A\} \quad (11)$$

Here, T_a is the set of valid attribute triples from the training dataset, while T'_a is the set of corrupted attribute triples (A is the set of attributes in G). The corrupted triples are used as negative samples by replacing the head entity with a random entity or the attribute with a random attribute value. Here, $f(t_a)$ is the plausibility score that based on the embedding of the head entity h, the embedding of the relationship r, and the vector representation of the attribute value that computed using the compositional function $f_a(a)$.

Joint Learning of Structure Embedding and Attribute Character Embedding We use the attribute character embedding $\mathbf{h_{ce}}$ to shift the structure embedding $\mathbf{h_{se}}$ into the same vector space by minimizing the following objective function J_{SIM}:

$$J_{SIM} = \sum_{h \in G_1 \cup G_2} [1 - \|\mathbf{h}_{se}\|_2 \, . \, \|\mathbf{h}_{ce}\|_2] \quad (12)$$

Here, $\|\mathbf{x}\|_2$ is the L2-Norm of vector $\mathbf{x}$. As a result, the structure embedding captures the similarity of entities between two KGs based on entity relationship while the attribute character embedding captures the similarity of entities based on attribute values. The overall objective function of the joint learning of structure embedding and attribute character embedding is:

$$J = J_{SE} + J_{CE} + J_{SIM} \quad (13)$$

4.4 Entity Alignment

The joint learning of structure embedding and attribute character embedding lets the similar entities from G_1 and G_2 to have a similar embeddings. Hence, the resulting embeddings can be used for entity alignment. We compute the following equation for entity alignment.

$$h_{map} = \underset{h_2 \in G_2}{\operatorname{argmax}} \|\mathbf{h}_1\|_2 \,.\, \|\mathbf{h}_2\|_2 \tag{14}$$

Given an entity $h_1 \in G_1$, we compute the similarity between h_1 and all entities $h_2 \in G_2$. $\langle h_1, h_{map} \rangle$ is the expected pair of aligned entities. We use a similarity threshold β to filter the pair entities that are too dissimilar to be aligned.

4.5 Triple Enrichment via Transitivity Rule

Even though the structure embedding implicitly learns the relationship transitive information, the explicit inclusion of this information increases the number of attributes and related entities for each entity which helps identify the similarity between entities. For example, given triples $\langle$`dbp:Emporium_Tower, :locatedIn, dbp:London`$\rangle$ and $\langle$`dbp:London, :country, dbp:England`$\rangle$, we can infer that `dbp:Emporium_Tower` has a relationship (i.e., `":locatedInCountry"`) with `dbp:England`. In fact, this information can be used to enrich the related entity `dbp:Emporium_Tower`. We treat the one-hop transitive relation as follows. Given transitive triples $\langle h_1, r_1, t \rangle$ and $\langle t, r_2, t_2 \rangle$, we interpret $r_1.r_2$ as a relation from head entity h_1 to tail entity t_2. Therefore, the relationship between these transitive triples is defined as $\mathbf{h_1} + (\mathbf{r_1}.\mathbf{r_2}) \approx \mathbf{t_2}$. The objective functions of the transitivity-enhanced embedding models are adapted from the Eq. 4 and Eq. 9 by replacing the relationship vector $\mathbf{r}$ with $\mathbf{r_1}.\mathbf{r_2}$.

5 Experiments

We evaluate our model on four real KGs including *DBpedia* (DBP) (Lehmann et al. 2015), *LinkedGeoData* (LGD) (Stadler et al. 2012), *Geonames* (GEO)[1], and YAGO (Hoffart et al. 2013). We run our proposed model to align entities of DBP with those of LGD, GEO, and YAGO, respectively. We compare the aligned entities found by our model with those in three ground truth datasets, **DBP-LGD**, **DBP-GEO**, and **DBP-YAGO**, which contain aligned entities[2] between DBP and LGD, GEO, and YAGO, respectively. We focus on the `LOCATION` entities in the LGD and GEO KGs since both KGs contain mainly geographical data. We consider `LOCATION`, `PERSON`, and `ORGANIZATION` entities in the YAGO KG. DBP-YAGO contains 15,000 aligned entities and 72 aligned predicates from a total of 279 predicates; DBP-LGD contains 10,000 aligned entities and ten aligned predicates from a total of 510 predicates; and DBP-GEO contains 10,000 aligned entities and ten aligned predicates from a total of 716 predicates. The statistics of the datasets are listed in Table 2.

[1]http://www.geonames.org/ontology/

[2]http://downloads.dbpedia.org/2016-10/links/

Table 2: Dataset Statistics

Dataset		Entities	Attribute Triples	Relationship Triples
DBP-LGD	LGD	24,309	90,054	10,084
	DBP	22,748	166,008	19,594
DBP-GEO	GEO	21,794	98,790	17,410
	DBP	22,748	166,008	19,594
DBP-YAGO	YAGO	30,628	173,309	38,451
	DBP	33,627	184,672	36,906

We use grid search to find the best hyperparameters for the models. We choose the embeddings dimensionality d among $\{50, 75, 100, 200\}$, the learning rate of the Adam optimizer among $\{0.001, 0.01, 0.1\}$, and the margin γ among $\{1, 5, 10\}$. We train the models with a batch size of 100 and a maximum of 400 epochs. We compare our proposed model with **TransE** (Bordes et al. 2013), **MTransE** (Chen et al. 2017a), and **JAPE** (Sun, Hu, and Li 2017).

5.1 Entity Alignment Results

We evaluate the performance of the models using $\mathbf{hits@k}(k = 1, 10)$ (i.e., the proportion of correctly aligned entities ranked in the top k predictions), and the mean of the rank (**MR**) of the correct (i.e., matching) entity. Higher $hits@k$ and lower MR indicate better performance. For each entity from DBP, we use Eq. 14 to compute the similarity scores with entities from the other KG (LGD/GEO/YAGO).

Table 3 shows that our proposed model consistently outperforms the baseline models, with $p < 0.01$ based on the t-test on the MR. As expected, TransE has poor performance on the entity alignment task because its embeddings of different KGs fall into different vector spaces, and hence it fails to capture the entity similarity between KGs. Meanwhile, MTransE and JAPE rely on the number of the seed alignments (we use 30% of the gold standard as the seed alignments as suggested in the original papers).

Among our attribute character embedding models, using the N-gram compositional function achieves better performance than using either the LSTM or the SUM compositional functions because the N-gram compositional function preserves string similarity better when mapping attribute strings to their vector representations than the other functions. The transitivity rule further improves the performance of the model since it enriches the attributes of the entities which allows more attributes to be used in the alignment.

To evaluate the power of our attribute character embedding in capturing the similarity between entities, we further create rule-based models for entity alignment, where we simply use the edit distance between entity label strings to align the entities. For the DBP-LGD and DBP-GEO datasets, we add coordinate similarity as an additional metric since both datasets only contain `LOCATION` entities. From Table 4, we can see that the resulting embeddings of our model can be added as an additional feature to enhance the performance of rule-based models.

Table 3: Entity Alignment Results

Model		DBP-LGD			DBP-GEO			DBP-YAGO		
		Hits@1	Hits@10	MR	Hits@1	Hits@10	MR	Hits@1	Hits@10	MR
Baselines	TransE	2.61	7.01	18445	1.34	6.71	17145	1.22	3.54	24809
	MTransE	33.29	34.32	10194	33.34	33.98	10240	33.46	34.32	7105
	JAPE	33.33	33.35	5104	33.35	33.75	5088	33.35	33.37	5296
Proposed	SUM	51.15	65.21	461	51.33	62.81	851	80.61	83.32	237
	LSTM	60.72	71.47	320	61.11	73.08	194	85.51	92.07	126
	N-gram	84.27	91.85	53	87.61	92.15	80	89.69	**95.83**	**23**
	Transitivity-enhanced models									
	SUM	52.01	65.61	352	54.05	65.29	712	81.12	84.22	171
	LSTM	61.75	73.86	148	62.29	74.23	232	86.65	92.91	82
	N-gram	**85.32**	**93.21**	**24**	**88.61**	**92.71**	**61**	**91.02**	95.59	26

Table 4: Rule-based Entity Alignment Results

Model	DBP-LGD			DBP-GEO			DBP-YAGO		
	Hits@1	Hits@10	MR	Hits@1	Hits@10	MR	Hits@1	Hits@10	MR
Label String	80.52	80.57	2603	79.32	79.41	2048	76.41	76.88	484
Label String + Coordinate	86.27	88.85	380	87.61	89.15	441	n/a	n/a	n/a
Label String + Embeddings	**88.91**	**89.12**	**28**	**91.51**	**91.56**	**23**	**86.42**	**86.63**	**66**

Table 5: KG Completion Results

Model	Link Prediction			Triple Classification		
	Hits@10			Precision		
	LGD	GEO	YAGO	LGD	GEO	YAGO
TransE	78.80	78.77	65.81	80.46	76.94	66.45
MTransE	65.55	63.89	60.40	77.41	73.81	63.21
JAPE	72.89	71.97	61.31	75.19	72.94	62.32
SUM	76.80	75.76	64.89	79.06	75.91	62.64
LSTM	76.69	76.89	63.75	79.44	76.41	64.98
N-gram	76.12	76.37	64.06	78.39	75.11	64.93
Transitivity-enhanced model						
SUM	74.87	74.31	62.51	78.21	74.43	62.75
LSTM	75.82	75.43	62.81	79.21	73.71	64.13
N-gram	75.69	76.55	63.40	78.14	74.21	64.73

5.2 Discussion

We further evaluate our proposed model on KG completion tasks. We follow two standard tasks including link prediction (Bordes et al. 2013) and triple classification (Socher et al. 2013). Link prediction aims to predict the missing entity (either the head entity h or the tail entity t) given a relationship r and an entity (i.e., predicting h given r and t; or predicting t given h and r). The evaluation protocol for link prediction is defined as follows. First, each relationship triple is corrupted by replacing its head or tail with all possible entities in the dataset. Then, these corrupted triples are ranked ascendingly based on the plausibility score $(\mathbf{h} + \mathbf{r} - \mathbf{t})$ (i.e., valid triples should have smaller plausibility scores than corrupted triples). We report $hits@10$ for the link prediction task. Triple classification aims to determine whether a triple $\langle h, r, t \rangle$ is a valid relationship triple or not. A binary classifier is trained based on the plausibility score $(\mathbf{h} + \mathbf{r} - \mathbf{t})$. Similar to link prediction, we create negative samples by corrupting the valid triples. We report the percentage of correctly classified triples.

Table 5 show the results of KG completion tasks. Despite not specifically designed for KG completion tasks, our proposed models achieve competitive performance on these tasks compared to TransE which was proposed for these tasks. This degradation in performance is due to that parameter α in our model guides the model to learn more on the aligned triples. However, the degradation is not significant with $p > 0.05$ based on the t-test on the MR of the link prediction results. Moreover, our method achieves better performance than the existing embedding-based entity alignment models MTransE and JAPE on these tasks.

6 Conclusion

We proposed an embedding model that integrates entity structure embedding with attribute character embedding for entity alignment between knowledge graphs. Our proposed model uses the attribute character embedding to shift the entity embeddings from different KGs to the same vector space. Moreover, we adopt the transitivity rule to enrich the number of attributes of an entity that helps identify the similarity between entities based on the attribute embeddings. Our proposed model outperforms the baselines consistently by over 50% in terms of $hits@1$ on alignment of entities between three pairs of real-world knowledge graphs.

Acknowledgments

Bayu Distiawan Trisedya is supported by the Indonesian Endowment Fund for Education (LPDP). This work is supported by Australian Research Council (ARC) Discovery Project DP180102050 and Future Fellowships Project FT120100832, Google Faculty Research Award, and the National Science Foundation of China (Project No. 61872070).

References

Bordes, A.; Weston, J.; Collobert, R.; and Bengio, Y. 2011. Learning structured embeddings of knowledge bases. In *Proceedings of AAAI Conference on Artificial Intelligence*, 301–306.

Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; and Yakhnenko, O. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of International Conference on Neural Information Processing Systems*, 2787–2795.

Chen, M.; Tian, Y.; Yang, M.; and Zaniolo, C. 2017a. Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. In *Proceedings of International Joint Conference on Artificial Intelligence*, 1511–1517.

Chen, M.; Zhou, T.; Zhou, P.; and Zaniolo, C. 2017b. Multi-graph affinity embeddings for multilingual knowledge graphs. In *Proceedings of NIPS Workshop on Automated Knowledge Base Construction*.

Fellbaum, C. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Hoffart, J.; Suchanek, F. M.; Berberich, K.; and Weikum, G. 2013. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence* 194:28–61.

Ji, G.; He, S.; Xu, L.; Liu, K.; and Zhao, J. 2015. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of International Joint Conference on Natural Language Processing*, 687–696.

Kim, Y.; Jernite, Y.; Sontag, D.; and Rush, A. M. 2016. Character-aware neural language models. In *Proceedings of AAAI Conference on Artificial Intelligence*, 2741–2749.

Lehmann, J.; Isele, R.; Jakob, M.; Jentzsch, A.; Kontokostas, D.; Mendes, P. N.; Hellmann, S.; Morsey, M.; van Kleef, P.; Auer, S.; and Bizer, C. 2015. Dbpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web* 6(2):167–195.

Lin, Y.; Liu, Z.; Sun, M.; Liu, Y.; and Zhu, X. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of AAAI Conference on Artificial Intelligence*, 2181–2187.

Ngomo, A.-C. N., and Auer, S. 2011. Limes: a time-efficient approach for large-scale link discovery on the web of data. In *Proceedings of International Joint Conference on Artificial Intelligence*, 2312–2317.

Nickel, M.; Rosasco, L.; and Poggio, T. A. 2016. Holographic embeddings of knowledge graphs. In *Proceedings of AAAI Conference on Artificial Intelligence*, 1955–1961.

Nickel, M.; Tresp, V.; and Kriegel, H.-P. 2012. Factorizing yago: scalable machine learning for linked data. In *Proceedings of International Conference on World Wide Web*, 271–280.

Niu, X.; Rong, S.; Wang, H.; and Yu, Y. 2012. An effective rule miner for instance matching in a web of data. In *Proceedings of International Conference on Information and Knowledge Management*, 1085–1094.

Pershina, M.; Yakout, M.; and Chakrabarti, K. 2015. Holistic entity matching across knowledge graphs. In *Proceedings of International Conference on Big Data*, 1585–1590.

Raimond, Y.; Sutton, C.; and Sandler, M. B. 2008. Automatic interlinking of music datasets on the semantic web. In *Linking Data on the Web Workshop. CEUR Workshop Proceedings, ISSN 1613-0073*.

Rivas, E., and Eddy, S. R. 1999. A dynamic programming algorithm for rna structure prediction including pseudoknots. *Journal of Molecular Biology* 285(5):2053–2068.

Scharffe, F.; Yanbin, F. L.; and Zhou, C. 2009. Rdf-ai: an architecture for rdf datasets matching, fusion and interlink. In *Proceedings of IJCAI Workshop on Identity and Reference in Knowledge Representation*.

Socher, R.; Chen, D.; Manning, C. D.; and Ng, A. Y. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Proceedings of International Conference on Neural Information Processing Systems*, 926–934.

Stadler, C.; Lehmann, J.; Hoffner, K.; and Auer, S. 2012. Linkedgeodata: A core for a web of spatial open data. *Semantic Web* 3(4):333–354.

Sun, Z.; Hu, W.; and Li, C. 2017. Cross-lingual entity alignment via joint attribute-preserving embedding. In *Proceedings of International Semantic Web Conference*, 628–644.

Trisedya, B. D.; Qi, J.; Zhang, R.; and Wang, W. 2018. Gtr-lstm: A triple encoder for sentence generation from rdf data. In *Proceedings of Association for Computational Linguistics*, 1627–1637.

Volz, J.; Bizer, C.; Gaedke, M.; and Kobilarov, G. 2009. Discovering and maintaining links on the web of data. In *Proceedings of International Semantic Web Conference*, 650–665.

Wang, Z., and Li, J. 2016. Text-enhanced representation learning for knowledge graph. In *Proceedings of International Joint Conference on Artificial Intelligence*, 1293–1299.

Wang, Z.; Zhang, J.; Feng, J.; and Chen, Z. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of AAAI Conference on Artificial Intelligence*, 1112–1119.

Xie, R.; Liu, Z.; Jia, J.; Luan, H.; and Sun, M. 2016. Representation learning of knowledge graphs with entity descriptions. In *Proceedings of AAAI Conference on Artificial Intelligence*, 2659–2665.

Yih, W.-t.; Chang, M.-W.; He, X.; and Gao, J. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of Association for Computational Linguistics*, 1321–1331.

Zhang, F.; Yuan, N. J.; Lian, D.; Xie, X.; and Ma, W.-Y. 2016. Collaborative knowledge base embedding for recommender systems. In *Proceedings of International Conference on Knowledge Discovery and Data Mining*, 353–362.

Zhu, H.; Xie, R.; Liu, Z.; and Sun, M. 2017. Iterative entity alignment via joint knowledge embeddings. In *Proceedings of International Joint Conference on Artificial Intelligence*, 4258–4264.