Knowledge Graph Alignment with Entity-Pair Embedding

Zhichun Wang, Jinjian Yang, Xiaoju Ye
School of Artificial Intelligence, Beijing Normal University, Beijing, China
Engineering Research Center of Intelligent Technology and Educational Application,
Ministry of Education, Beijing, China
zcwang@bnu.edu.cn {tecyyang,mryx}@mail.bnu.edu.cn

Abstract

Knowledge Graph (KG) alignment is to match entities in different KGs, which is important to knowledge fusion and integration. Recently, a number of embedding-based approaches for KG alignment have been proposed and achieved promising results. These approaches first embed entities in low-dimensional vector spaces, and then obtain entity alignments by computations on their vector representations. Although continuous improvements have been achieved by recent work, the performances of existing approaches are still not satisfactory. In this work, we present a new approach that directly learns embeddings of entity-pairs for KG alignment. Our approach first generates a pair-wise connectivity graph (PCG) of two KGs, whose nodes are entity-pairs and edges correspond to relation-pairs; it then learns node (entity-pair) embeddings of the PCG, which are used to predict equivalent relations of entities. To get desirable embeddings, a convolutional neural network is used to generate similarity features of entity-pairs from their attributes; and a graph neural network is employed to propagate the similarity features and get the final embeddings of entity-pairs. Experiments on five real-world datasets show that our approach can achieve the state-of-the-art KG alignment results.

1 Introduction

Knowledge graphs (KGs) have been built and applied in several domains, including question answering (Zhang et al., 2018), recommendation (Sun et al., 2018b), and information extraction (Yang and Mitchell, 2017). Most existing KGs are built separately by different organizations, using different data sources and languages. Therefore, KGs are heterogeneous that the same entity may exist in different KGs in different surface forms. On the other hand, KGs can be complementary to each other; knowledge about the same entity may distribute in several KGs. To handle the heterogeneity problem and integrate knowledge in different KGs, it is essential to perform KG alignment, i.e. matching entities in separate KGs.

Recently, KG embedding models have been explored in solving the problem of KG alignment. A number of embedding-based approaches have been proposed, including MTransE (Chen et al., 2017), JAPE (Sun et al., 2017), IPTransE (Zhu et al., 2017), GCN-Align (Wang et al., 2018), RDGCN (Wu et al., 2019), and MultiKE (Zhang et al., 2019), etc. These approaches first embed entities in low-dimensional vector spaces, and then obtain the entity alignments by computations on their vector representations. Comparing with traditional similarity-based approaches, embedding-based ones can effectively model different kinds of information in KGs, which align entities without manually designed similarity features. Most recently, continuous improvements have been achieved by combining multiple kinds of information in KGs or using more sophisticated embedding models. However, the performances of most approaches are still not satisfactory. According to the results in a recent work (Zhang et al., 2019), a traditional unsupervised alignment approach, Logmap (Jiménez-Ruiz and Cuenca Grau, 2011), outperforms most existing embedding-based approaches. To get more accurate alignment results, we propose an entity-pair embedding approach for KG alignment (EPEA). Instead of learning embeddings of single entities, our approach directly learns representations of entity-pairs. Similarity features of entities' attribute information are automatically extracted, which are then propagated using structure information of entities. Equivalent relations of entities can be accurately predicted based on the learned embeddings of entity-pairs.

Specifically, our work has the following contri-

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 1672–1680,
November 16–20, 2020.

butions:

- We introduce the definition of pairwise connectivity graph (PCG) of KGs, whose nodes are entity-pairs and edges correspond to relation-pairs. We solve the KG alignment problem via node embedding of the PCG.
- We propose a similarity feature extraction method based on convolutional neural network (CNN), which automatically generates feature vectors of entity-pairs encoding their attribute similarities.
- We propose a graph neural network (GNN) with edge-aware attentions to propagate similarity features in the PCG. Similarity features are propagated among the neighbors of entity-pairs, which incorporate structure similarity into the embeddings of entity-pairs.
- In the experiments on aligning real-world KGs, our approach outperforms the compared approaches, and achieves the state-of-the-art results.

The rest of this paper is organized as follows: Section 2 formalizes the entity alignment problem, Section 3 describes our proposed approach, Section 4 presents the evaluation results, Section 5 discusses some related work, and Section 6 is the conclusion.

2 Problem Formulation

2.1 KG and KG Alignment

KGs represent structural information about entities in real-world as triples having the form of $\langle s, p, o \rangle$. In this work, our KG alignment model considers both relational and attributional triples in KGs. The relational triples describe relations between entities, and the attributional triples describe attributes of entities. We formally represent a KG as $G = (E, R, A, L, T)$, where E, R, A and L are sets of entities, relations, attributes, and literals; $T \subseteq (E \times R \times E) \cup (E \times A \times L)$ is the sets of triples. Given two KGs $G = (E, R, A, L, T)$ and $G' = (E', R', A', L', T')$, the task of KG alignment is to find, for each entity in E, the equivalent entity in E'.

2.2 Pair-wise Connectivity Graph

Pair-wise connectivity graph (PCG) can capture interactions of node-pairs of two directed graphs (Wang et al., 2012; Melnik et al., 2002). In this work, we define the PCG of KGs. For two KGs, each node in their PCG corresponds to an entity-pair from two KGs, and each edge connecting two nodes reflects the correlation between two entity-pairs. By generating the PCG of two KGs, the problem of KG alignment is then transformed to node embedding and classification (i.e. equivalent or nonequivalent) in the PCG. For two KGs $G = (E, R, A, L, T)$ and $G' = (E', R', A', L', T')$, the PCG of them is $\mathcal{G}(G, G') = (\mathcal{E}, \mathcal{R}, \mathcal{T})$, where $\mathcal{E}$, $\mathcal{R}$ and $\mathcal{T}$ are sets of nodes, edge types and edges. Each element in $\mathcal{E}$ corresponds to an entity-pair between G and G', and each element in $\mathcal{R}$ corresponds to an relation-pair. $\mathcal{T}$ is a set of typed edges between nodes, each edge is established as follows:

$$\begin{aligned} &\langle a, r, b \rangle \in T \wedge \langle a', r', b' \rangle \in T' \\ &\Longleftrightarrow \langle (a, a'), (r, r'), (b, b') \rangle \in \mathcal{T} \end{aligned} \tag{1}$$

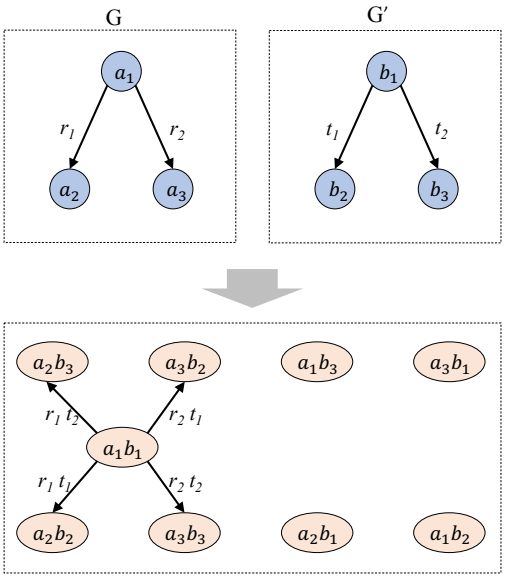


Figure 1: Pair-wise connectivity graph.

Figure 1 shows an example of PCG of two KGs. There are two KGs, each of them has three entities. The PCG of them contains nine nodes representing all the possible entity-pairs of two KGs; and there are four typed edges in the PCG. PCG can represent the connections of entity-pairs between two KGs, we use PCG to capture the interaction of possible entity alignments between two KGs. In our approach, the problem of KG alignment will be solved via node embedding of the PCG. Equivalent relations of entities are predicted based on the learned embeddings.

3 The Proposed Approach

Figure 2 shows the framework of our approach. Given two KGs, our approach first generates the PCG of them. Then, a CNN-based feature extraction method is used to generate node representations from the attribute information of entities. At last, an attention-based feature propagation is performed over the PCG to incorporate structure information into the node representations. Entity alignments are predicted based on the learned embeddings of entity-pairs. In the following, we present our approach in detail.

3.1 Generating the PCG

To generate the PCG of two KGs, we can first pair all the entities from two KGs as nodes, and then use Equation 1 to generate edges between nodes. However, KGs usually contain large number of entities, the PCG of two large-scale KGs will contain huge number of nodes. To avoid pairing all the entities from two KGs and control the size of the PCG, our approach selects entity-pairs having high equivalent possibilities as nodes in the PCG. Specifically, Locality-Sensitive Hashing (LSH) is employed in our approach to efficiently find similar entities between two KGs. LSH hashes similar items more likely into the same bucket than dissimilar items. Before using LSH, our approach first uses one of the following methods to generate set-representations of entities, which are used in the hashing process.

- **N-grams of Names.** If entity names are available and in the same language, this method generates a set of character-level n-grams of entities' names as the set-representations of entities.
- **N-grams of Attributes.** This method treats attribute values of an entity as text strings, and generates character-level n-grams of all the attribute values for each entity. All the n-grams are then merged into a set as the representation of the entity.
- **Seeding alignments.** If seeding alignments between two KGs are available, a set of aligned entities in an entity's neighborhood will be taken as the set-representation.

After being represented as sets of elements (n-grams or neighboring entities) by one of the above methods, all the entities in two KGs are hashed using LSH. To select entity-pairs as nodes in the PCG of G and G', our approach efficiently finds, for each entity $e \in G$, a set of entities $C_e = \{e' | e' \in G', J(e, e') > \delta\}$ as its alignment candidates, where $J(e, e')$ is the Jaccard similarity of two entities, δ is a predefined threshold. Entity e is then paired with all the entities in C_e to form the nodes in the PCG.

3.2 Attribute Feature Generation

Entities having the same or similar attribute values tend to be equivalent. Therefore, comparing attribute values of two entities are important for discovering entity alignments. In traditional approaches, attributes have to be first matched manually, then the values of corresponding attributes can be compared to get similarities between entities. In some of the embedding-based approaches, attribute types or values are utilized to generate attribute embeddings, which are integrated with structure embeddings of entities to get more accurate entity alignments. In this work, we extract similarity features from entities' attributes in an automatic way.

CNN-based Feature Extraction

We propose an attribute feature extraction method based on Convolutional Neural Network (CNN). Our method can automatically obtain useful similarity features of entity-pairs without any human effort. It generates a vector representation of each entity-pair in the PCG, which captures attribute similarities of two entities.

Given an entity-pair (e, e'), where $e \in G$ and $e' \in G'$. Let $A = \{A_1, ..., A_n\}$ and $A' = \{A'_1, ..., A'_m\}$ be two sets of all the attributes in G and G', respectively. Let $A_i(e)$ denotes the value of the i-th attribute of e, $A'_j(e')$ denotes the value of the j-th attribute of e'. To capture various similarities between two entities e and e', a similarity matrix $\mathbf{M}_{m \times n}$ is computed by comparing values of every attribute pair of two entities. Each element m_{ij} in $\mathbf{M}$ is the similarity of $A_i(e)$ and $A'_j(e')$. Attribute values in KGs may have various types, for example data, time, float, integer and string. To keep simplicity and effectiveness, our approach treats all the attribute values as strings. Similarities of attribute values are computed as N-gram-based Jaccard similarities of strings:

$$Jaccard(s, t) = \frac{|NG(s) \cap NG(t)|}{|NG(s) \cup NG(t)|} \quad (2)$$

where $NG(s)$ and $NG(t)$ are n-grams of strings s

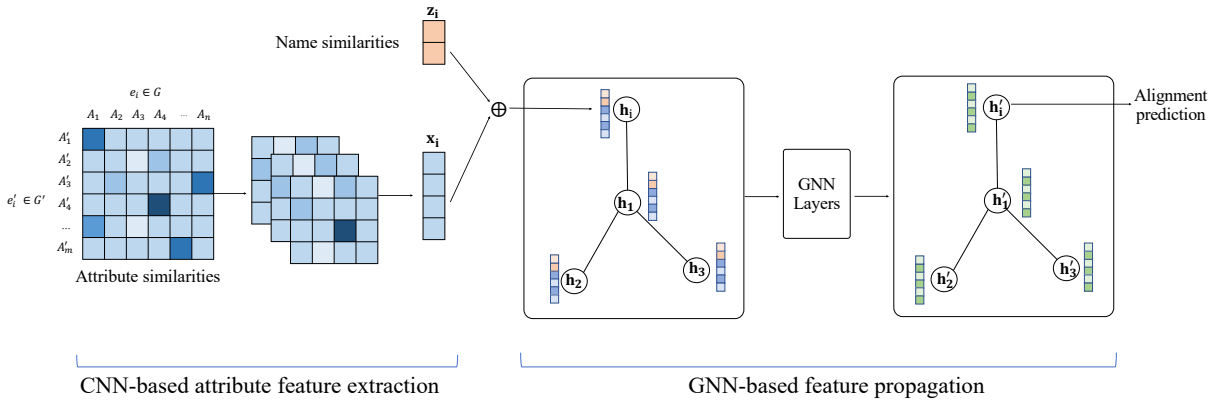


Figure 2: Framework.

and t.

Usually, one entity is only described by a small number of attributes in a KG. Therefore, for an entity, the values of many attributes are empty. The similarity matrix of two entities is usually a sparse one, with a large proportion of 0s in it. Meanwhile, similarities between some attributes may be useless for detecting alignments. To automatically find useful similarity patterns of attribute values, we use a CNN model to encode the sparse similarity matrix into a short and dense vector.

The input of the CNN is the similarity matrix $\mathbf{M}$ of two entities, two convolution layers are used to generate a dense similarity vector from $\mathbf{M}$. For the l-th convolution layer, its output is computed as follows:

$$\mathbf{X}_k^{(l)} = \text{ReLU}\left(\mathbf{W}_k^{(l)} \otimes \mathbf{X}^{(l-1)} + \mathbf{b}_k^{(l)}\right) \quad (3)$$

where $\mathbf{X}^{(l-1)}$ is the input of l-th layer; for the first layer, $\mathbf{X}^{(0)} = \mathbf{M}$; we use multiple filters to extract useful similarity features from the input, $\mathbf{W}_k^{(l)}$ is the k-th filter of l-th layer, $\mathbf{b}_k^{(l)}$ is the bias of the k-th filter in l-th layer; $\otimes$ is the convolution operator. There is a max pooling layer after each convolution layer. The output features of last max pooling layer is the similarity vector of the entity-pair.

Name Similarity Features

In this work, name or label of an entity is considered as a special attribute, which is an important clue for determining whether two entities are equivalent. If entities' names are available in KGs, our approach computes a name similarity vector for each entity-pair, which will be concatenated with the similarity vector generated by the CNN model. To capture similarity features of entities' names from different aspects, we use multiple string-based similarity metrics, which are widely used in traditional similarity-based alignment approaches. If entities' names are in different languages in two KGs, machine translation tool will be used to translate names in one language to the other language. Let s and t be names of two entities, the following similarity measures are used in our approach.

- **String equality.** It measures whether two strings are the same:

$$z_1(s,t) = \begin{cases} 1 & \text{if } s = t, \\ 0 & else. \end{cases} \quad (4)$$

- **Edit Distance.** It evaluates the minimal cost of operations which have to applied to one of the strings to obtain the other string:

$$z_2(s,t) = 1 - \frac{|\{ops\}|}{max(len(s), len(t))}, \quad (5)$$

where $\{ops\}$ denotes the set of operations, $len(\cdot)$ is the string length.

- **Jaccard Similarity.** It computes the Jaccard Similarity of the character-level n-grams of two strings, as defined in Equation 2, we denote this similarity as $z_4(s,t)$.

- **Substring Similarity.** It is computed by finding the longest common substring of two strings.

$$z_4(s,t) = \frac{2|LCS(s,t)|}{|s| + |t|}, \quad (6)$$

where $LCS(s,t)$ is the longest common substring of s and t.

Let $\mathbf{z} = [z_1, z_2, z_3, z_4]$ denote name similarities of an entity-pair, it will be concatenated with the similarity vector $\mathbf{x}$ generated by CNN to form the initial feature vector of the entity-pair. The feature vectors of all the entity-pairs will be passed to an attention-based propagation process, to generate the final embeddings.

3.3 Attention-based Feature Propagation

Equivalent entities in two KGs are usually neighbored by some other equivalent entities. Therefore, structure information in KGs are very important for discovering entity alignments. In our work, edges between nodes in the PCG reflect the neighboring information of entity-pairs. To obtain feature representations of entity-pairs containing their neighbors' information, our approach propagates attribute features of entity-pairs following these edges. Specifically, our approach uses a Graph Neural Network (GNN) to propagate the attribute features of entity-pairs over the PCG. GNNs learn node representations in a graph by recursively aggregating the feature vectors of its neighbors, which are able to combine the node features and structure information in the graph. Several approaches have exploited GNNs for embedding-based KG alignment, which achieved promising results. In the previous approaches, GNNs are used for learning representations of entities. While in this work, we design a new GNN model for learning vector representations of entity-pairs.

Our model is a residual GNN with edge-aware attentions, which is built by modifying the attention mechanism of the GAT model (Velickovic et al., 2017). Our GNN model has two layers, each layer takes a set of node features $\mathbf{H} = \{\mathbf{h_1}, \mathbf{h_2}, ..., \mathbf{h_N}\}$ as inputs, where $\mathbf{h_i} \in \mathbb{R}^F$ and N is the number of nodes in the PCG, F is the dimension of the input features. Each layer generates a new set of node representations $\mathbf{H'} = \{\mathbf{h'_1}, \mathbf{h'_2}, ..., \mathbf{h'_N}\}$, $\mathbf{h'_i} \in \mathbb{R}^{F'}$ and it is computed as:

$$\mathbf{h'_i} = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W h_j}\right), \tag{7}$$

where $\mathcal{N}_i$ is the set of neighboring nodes of the i-th node (ignoring the edge directions in the PCG), $\mathbf{W} \in \mathbb{R}^{F \times F'}$ is a shared matrix, α_{ij} is a learnable attention indicating the importance of the j-th node to the i-th node.

Edge-aware Attention Mechanism

In the GAT model, the attention α_{ij} is computed based on the features of node i and j. In the task of KG alignment, we consider that the type of edge between two nodes is important and should not be ignored. Therefore, we use an edge-aware attention mechanism to compute the attention α_{ij}. A shared attentional mechanism $\mathbb{R}^{F'} \times \mathbb{R}^{F'} \times \mathbb{R}^{F'} \rightarrow \mathbb{R}$ is used to computes attention coefficients:

$$e_{ij} = \text{LeakyReLU}\left(\mathbf{a}^\top \left[\mathbf{W h_i} \| \mathbf{W h_j} \| \mathbf{t_{(i \rightarrow j)}}\right]\right) \tag{8}$$

where $(i \rightarrow j)$ denotes the index of edge-type linking the i-th node to the j-th node, $\mathbf{t_{(i \rightarrow j)}} \in \mathbb{R}^{F'}$ is the vector representation of the edge-type; $\mathbf{a} \in \mathbb{R}^{3F'}$ is a weight vector of a single-layer feed-forward neural network for computing the attention coefficients; $\|$ represents concatenation of vectors. Here the vector of an edge-type is computed based on the nodes' vectors connected by it. For an edge-type t_k, let S_k and T_k be the sets of nodes' indices having outgoing edges and incoming edges of the type in the PCG respectively, the vector representation of t_k is computed as:

$$\mathbf{t_k} = |\frac{1}{|S_k|} \sum_{i \in S_k} \mathbf{W h_i} - \frac{1}{|T_k|} \sum_{j \in T_k} \mathbf{W h_j}|, \tag{9}$$

which is the element-wise absolute difference between the mean vectors of source and target nodes connected by t_k.

When the attention coefficients are obtained following Equation 8, normalized attentions are then computed using a softmax function over all the coefficients of its neighboring nodes:

$$a_{ij} = \text{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})} \tag{10}$$

where $\mathcal{N}_i$ is the set of neighboring nodes of the i-th node.

Residual Connections in GNN

To let the entity-pair embeddings memorize the original attribute features, we add residual connections from the input features to the output layer of the GNN model. We let $F = F'$, i.e. the sizes of input and output node vectors of each GNN layer are the same. A shortcut connection between the input and output layers is added, and the final representation of a node is computed by element-wise addition of $\mathbf{h_i^0}$ and $\mathbf{h_i^L}$, where $\mathbf{h_i^0} = [\mathbf{x_i} || \mathbf{z_i}]$ and $\mathbf{h_i^L}$ are the input and output features of the i-th node.

3.4 Model Training

There are two neural network models in our approach, i.e. the CNN model for attribute feature extraction and the GNN model for feature propagation. These two separate models are trained sequentially, using the same training data. For two KGs G and G', let $A = \{(e_i, v_i)|e_i \in G, v_i \in G'\}_{i=1}^{K}$ be a set of known entity alignments, they will be used as training data for both models.

For the CNN model, let $\mathbf{x_i}$ be the attribute feature vector of entity-pair (e_i, v_i) generated by the model. We use one fully-connected layer to generate a score for each entity-pair, taking $\mathbf{x_i}$ as the input:

$$S_{CNN}(e_i, v_i) = \sigma(\mathbf{c}^\top \mathbf{x_i} + \alpha) \tag{11}$$

where $\mathbf{c} \in \mathbb{R}^d$ and $\alpha \in \mathbb{R}$ are parameters, σ is the Sigmoid function.

For the GNN model, let $\mathbf{h_i}$ be the feature vector of entity-pair (e_i, v_i) after the feature propagation with the model. A similar score function is also defined as:

$$S_{GNN}(e_i, v_i) = \sigma(\mathbf{g}^\top \mathbf{h_i} + \beta) \tag{12}$$

where $\mathbf{g} \in \mathbb{R}^d$ and $\beta \in \mathbb{R}$ are parameters, σ is also the Sigmoid function.

For both models, we want the aligned entity-pairs having higher scores than the non-aligned entity-pairs. Therefore, two models are both trained by minimizing the following margin-based ranking loss function:

$$\mathcal{L} = \sum_{(e,v)\in A} \sum_{(e',v')\in A'_{(e,v)}} [\gamma - S(e,v) + S((e',v'))]_+ \tag{13}$$

where $[x]_+ = max\{0, x\}$, $\gamma > 0$ is a margin hyper-parameter, $A'_{(e,v)}$ denotes the set of non-aligned entity-pairs in the PCG containing entity e or v. The score S is either S_{CNN} or S_{GNN}, depending on which model is trained.

4 Experiments

4.1 Datasets

Five datasets are used to evaluate our approach, each dataset contains two knowledge graphs to be aligned. Table 1 outlines the detail information of these datasets. $\mathrm{DBP15K_{ZH-EN}}$, $\mathrm{DBP15K_{JA-EN}}$ and $\mathrm{DBP15K_{FR-EN}}$ were built by (Sun et al., 2017). They are generated from DBpedia and each dataset contains 15 thousand aligned entity pairs in two language versions of DBpedia. DBP-WD and DBP-YG were first used in (Sun et al., 2018a), which are generated from DBpedia, Wikidata and YAGO3. Each dataset contains 100 thousand aligned entity pairs. For all the datasets, we use the same training/testing split of aligned entity pairs with previous work (Sun et al., 2017, 2018a), 30% for training and 70% for testing.

Table 1: Details of the datasets

Datasets		# Entities	# Relations	# Attributes
$\mathrm{DBP_{ZH-EN}}$	Chinese	66,469	2,830	8,113
	English	98,125	2,317	7,173
$\mathrm{DBP_{JA-EN}}$	Japanese	65,744	2,043	5,882
	English	95,680	2,096	6,066
$\mathrm{DBP_{FR-EN}}$	French	66,858	1,379	4,547
	English	105,889	2,209	6,422
DBP-WD	DBpedia	100,000	330	351
	Wikidata	100,000	220	729
DBP-YG	DBpedia	100,000	302	334
	YAGO3	100,000	31	23

4.2 Experiment Settings

We implement our approach by using TensorFlow[1], and run experiments on a workstation with Intel Xeon 2.1GHz CPU, an NVIDIA Tesla P100 GPU and 64 GB memory. We use Hits@k and MRR(Mean reciprocal ranking) as the evaluation metrics, which are popular and widely used in other KG alignment work. Hits@k measures the percentage of correctly alignments ranked in the top k candidates. MRR is the average of the reciprocal ranks of the results. The higher Hits@k and MRR, the better is the performance. The dimensions of similarity features and final embeddings of entity-pairs are set to the same value, which is among {30, 60, 100, 120}, we consider the learning rate in two models among {0.1, 0.01, 0.002, 0.001}, the margin γ in loss functions among {1, 2, 4, 10}. Best configurations for two models in our approach are selected based on the MRR.

We compare our approach EPEA with recent KG alignment models, which can be divided into two groups. Models in the first group only use structure information in KGs, including MTransE (Chen et al., 2017), IPTransE (Zhu et al., 2017), BootEA (Sun et al., 2018a), MuGNN (Cao et al., 2019), RDGCN (Wu et al., 2019), AliNet (Zequn Sun, 2020), and NAEA (Zhu et al., 2019). The

[1]https://www.tensorflow.org/

Table 2: Results of KG alignment

Approaches	DBP_{ZH-EN}			DBP_{JA-EN}			DBP_{FR-EN}			DBP − WD			DBP − YG		
	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR
MTransE	0.308	0.614	0.364	0.279	0.575	0.349	0.244	0.556	0.335	0.281	0.520	0.363	0.252	0.493	0.334
IPTransE	0.406	0.735	0.516	0.367	0.693	0.474	0.333	0.685	0.451	0.349	0.638	0.447	0.297	0.558	0.386
BootEA	0.629	0.848	0.703	0.622	0.854	0.701	0.653	0.874	0.731	0.748	0.898	0.801	0.761	0.894	0.808
MuGNN	0.494	0.844	0.611	0.501	0.857	0.621	0.495	0.870	0.621	0.616	0.897	0.714	0.741	0.937	0.810
RDGCN	0.708	0.846	0.746	0.767	0.895	0.812	0.886	0.957	0.911	-	-	-	-	-	-
AliNet	0.539	0.826	0.628	0.549	0.831	0.645	0.552	0.852	0.657	0.690	0.908	0.766	0.786	0.943	0.841
NAEA	0.650	0.867	0.720	0.641	0.872	0.718	0.673	0.894	0.752	0.767	0.917	0.817	0.778	0.912	0.821
JAPE	0.412	0.745	0.490	0.363	0.685	0.476	0.324	0.667	0.430	0.318	0.589	0.411	0.236	0.484	0.320
GCN-Align	0.413	0.744	0.549	0.399	0.745	0.546	0.375	0.745	0.532	0.506	0.772	0.600	0.597	0.838	0.682
MultiKE	-	-	-	-	-	-	-	-	-	0.914	0.951	0.928	0.880	0.953	0.906
CEA	0.787	-	-	0.863	-	-	**0.972**	-	-	**0.998**	-	-	**0.999**	-	-
CNN	0.612	0.840	0.694	0.569	0.820	0.657	0.777	0.930	0.833	0.840	0.986	0.897	0.780	0.975	0.854
CNN+GAT	0.726	0.916	0.803	0.764	0.936	0.836	0.758	0.960	0.839	0.945	0.967	0.955	0.980	0.999	0.988
EPEA	**0.885**	**0.953**	**0.911**	**0.924**	**0.969**	**0.942**	0.955	**0.986**	**0.967**	0.975	**0.981**	**0.977**	**1.000**	**1.000**	**1.000**

other group of models use attribute or name information in KGs, including JAPE (Sun et al., 2017), GCN-Align (Wang et al., 2018), MultiKE (Zhang et al., 2019), and CEA (Zeng et al., 2020).

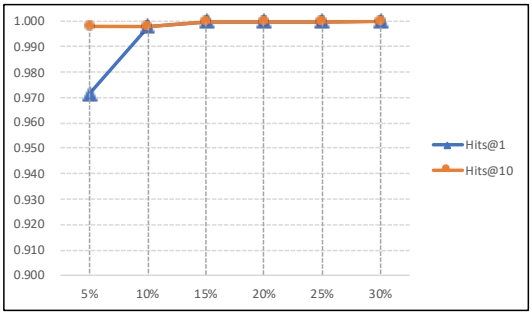
4.3 Results

Overall Comparisons. Table 2 shows the results of all approaches. Because all the approaches use the same sets of seeding and testing alignments in each dataset, the results of the compared approaches are obtained from their original papers. It shows that our approach EPEA achieves promising improvements compared with the previous approaches. Our approach outperforms all the compared approaches other than CEA on five datasets, in terms of Hits@1, Hits@10 and MRR. Taking no account of CEA, RDGCN achieved the state-of-the-art results on three cross-lingual datasets. Compared with RDGCN, our approach gets improvements of 17.7%, 15.7%, and 6.9% of Hits@1 on these datasets. MultiKE performed the best on DBP-WD and DBP-YG among the compared approaches excluding CEA, our approach outperforms MultiKE by 6.1% and 12.0% of Hits@1 on the two datasets, respectively. CEA is a strong approach which uses a collective alignment framework with adaptive feature fusion mechanism; only results of Hits@1 (i.e. accuracy) are reported by its authors. In terms of Hits@1, CEA performs better than RDGCN and MultiKE on cross-lingual and monolingual datasets, respectively. Compared with CEA, our approach gets higher Hits@1 on DBP_{ZH-EN} and DBP_{JA-EN}, and gets better results than CEA on DBP-YG. CEA performs better than EPEA on DBP_{FR-EN} and DBP-WD, but the results of two approaches are close, with small differences of 1.7% and 2.3%.

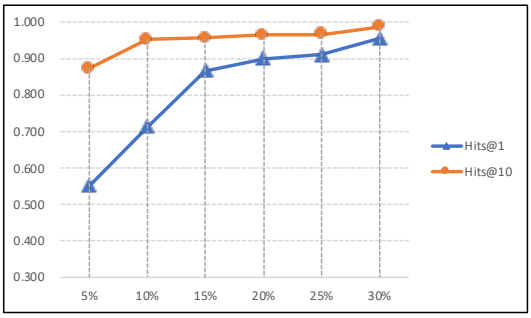
Contributions of component models. To analyze the contributions of component models in our approach, we build two variations of EPEA by removing or replacing GNN model. The first variation of EPEA is represented as CNN, which only uses the CNN model to predict alignments based on attribute features. The second variation of EPEA is represented as CNN+GAT, which replaces the edge-aware attentional GNN with GAT (Velickovic et al., 2017) in EPEA. The results of CNN and CNN+GAT are also outlined in Table 2. It shows that two sub-models of EPEA are both effective and important for the promising performance of EPEA. First, the CNN model can extract useful similarity features for predicting entity alignments, which gets better results than half of the comparison approaches, including MTransE, MuGNN, AliNet, JAPE, et al. Second, GNN-based feature propagation improves the results significantly, and the new designed GNN model edge-aware attention in EPEA works better than GAT. There is 11.9% improvements of Hits@1 on average when GAT is used to propagate the similarity features, while our new GNN model gets even bigger improvements, average 24.9% of Hits@1.

Impact of Seed Alignments. To investigate how the size of seed alignments (pre-aligned entity pairs for training) affects the results of our approach, we run our approach with different number of seed alignments. The proportions of seed alignments ranges from 5% to 30% with step of 5%. Figure 3

shows the $Hits@1$ and $Hits@10$ of EPEA on two datasets, DBP-YG and DBP$_{FR-EN}$. It shows that EPEA gets nearly optimal Hits on DBP-YG using 10% seed alignments, both Hits@1 and Hits@10 are 100% when more than 15% seed alignments are used. This is because DBP-YG contains rich attribute information of entities including entities' names, our approach can fully utilize attribute and structure information to accurately predict entity alignments even with small number of seed alignments. On the DBP$_{FR-EN}$ dataset, our approach gets >70% Hits@1 and >95% Hits@10 when only 10% seed alignments are used; it outperforms most of the compared approaches in Table 2 which use 30% seed alignments. As the number of seed alignments increases, our approach steadily improves the alignment results.



(a) DBP-YG



(b) DBP$_{FR-EN}$

Figure 3: Results of EPEA using different sizes of seed alignments (horizontal coordinates: proportions of pre-aligned entities used in training data; vertical coordinates: $Hits@k$)

5 Related Work

A number of embedding-based entity alignment approaches have been proposed recently. Some approaches mainly rely on the structure information in KGs to find alignments, including MTransE (Chen et al., 2017), IPTransE (Zhu et al., 2017), BootEA (Sun et al., 2018a), MuGNN (Cao et al., 2019), NAEA (Zhu et al., 2019), RDGCN (Wu et al., 2019) and AliNet (Zequn Sun, 2020). Entity embeddings are learned by using information of entity and their relations. MTransE encodes structure information of KGs in separate spaces, and then performs transitions from one space to the other. TPTransE and BootEA both are iterative alignment approaches, which use new discovered alignments to expand the seeding alignments. MuGNN employs a multi-channel GNN to learn alignment-oriented KG embeddings. NAEA enhances the TransE model by learning embeddings by a neighborhood-aware attentional representation method. RDGCN uses a relation-aware dual-graph convolutional network to incorporate relation information via attentive interactions between KG and its dual relation counterpart. AliNet is a GNN-based model which aggregates both direct and distant neighborhood information.

To get improved results, some approaches utilize entity attributes or names in KGs. JAPE (Sun et al., 2017) performs attribute embedding by Skip-Gram model which captures the correlations of attributes in KGs. GCN-Align (Wang et al., 2018) encodes attribute information of entities into their embeddings by using GCNs. MultiKE (Zhang et al., 2019) uses a framework unifying the views of entity names, relations and attributes to learn embeddings for aligning entities. CEA (Zeng et al., 2020) combines structural, semantic and string features of entities, which are integrated with dynamically assigned weights.

Compared with the previous approaches, ours directly learns embeddings of entity-pairs, instead of entities. Attribute and structure information are encoded in the embeddings sequentially, and experiments validate the effectiveness of our approach.

6 Conclusion

This paper presents a new entity-pair embedding approach for KG alignment. Our approach first extracts useful attribute features of entity-pairs by using a convolutional neural network, and then propagates the features among the neighbors of entity-pairs, by using a graph neural network with edge-aware attentions. The embeddings are learned with the object of separating equivalent and nonequivalent entity-pairs. Experiments on five real-world datasets show that our approach achieves the state-of-the-art results.

Acknowledgments

The work is supported by the National Key Research and Development Program of China (No. 2017YFB1402105).

References

Yixin Cao, Zhiyuan Liu, Chengjiang Li, Zhiyuan Liu, Juanzi Li, and Tat-Seng Chua. 2019. Multi-channel graph neural network for entity alignment. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL2019)*, pages 1452–1461.

Muhao Chen, Yingtao Tian, Mohan Yang, and Carlo Zaniolo. 2017. Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (AAAI2017)*, pages 1511–1517.

Ernesto Jiménez-Ruiz and Bernardo Cuenca Grau. 2011. Logmap: Logic-based and scalable ontology matching. In *Proceedings of the Tenth International Semantic Web Conference(ISWC2011)*, pages 273–288.

S. Melnik, H. Garcia-Molina, and E. Rahm. 2002. Similarity flooding: a versatile graph matching algorithm and its application to schema matching. In *Proceedings 18th International Conference on Data Engineering (ICDE2002)*, pages 117–128.

Zequn Sun, Wei Hu, and Chengkai Li. 2017. Cross-lingual entity alignment via joint attribute-preserving embedding. In *Proceedings of the Sixteenth International Semantic Web Conference (ISWC2017)*, pages 628–644.

Zequn Sun, Wei Hu, Qingheng Zhang, and Yuzhong Qu. 2018a. Bootstrapping entity alignment with knowledge graph embedding. In *Proceedings of the Twenty-Seventh international joint conference on Artificial Intelligence (IJCAI2018)*, pages 4396–4402.

Zhu Sun, Jie Yang, Jie Zhang, Alessandro Bozzon, Long-Kai Huang, and Chi Xu. 2018b. Recurrent knowledge graph embedding for effective recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems(RecSys2018)*, pages 297–305.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2017. Graph attention networks. *CoRR*, abs/1710.10903.

Zhichun Wang, Juanzi Li, Zhigang Wang, and Jie Tang. 2012. Cross-lingual knowledge linking across wiki knowledge bases. In *Proceedings of the 21st International Conference on World Wide Web (WWW2012)*, pages 459–468.

Zhichun Wang, Qingsong Lv, Xiaohan Lan, and Yu Zhang. 2018. Cross-lingual knowledge graph alignment via graph convolutional networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP2018)*, pages 349–357.

Yuting Wu, Xiao Liu, Yansong Feng, Zheng Wang, Rui Yan, and Dongyan Zhao. 2019. Relation-aware entity alignment for heterogeneous knowledge graphs. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI2019)*, pages 5278–5284.

Bishan Yang and Tom Mitchell. 2017. Leveraging knowledge bases in LSTMs for improving machine reading. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL2017)*, pages 1436–1446.

W. Zeng, X. Zhao, J. Tang, and X. Lin. 2020. Collective entity alignment via adaptive features. In *2020 IEEE 36th International Conference on Data Engineering (ICDE2020)*, pages 1870–1873.

Wei Hu Muhao Chen Jian Dai Wei Zhang Yuzhong Qu Zequn Sun, Chengming Wang. 2020. Knowledge graph alignment network with gated multi-hop neighborhood aggregation. In *Proceedings of the Twenty-Nineth International Joint Conference on Artificial Intelligence (AAAI2020)*.

Qingheng Zhang, Zequn Sun, Wei Hu, Muhao Chen, Lingbing Guo, and Yuzhong Qu. 2019. Multi-view knowledge graph embedding for entity alignment. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI2019)*, pages 5429–5435.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, pages 6069–6076.

Hao Zhu, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2017. Iterative entity alignment via joint knowledge embeddings. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI2017)*, pages 4258–4264.

Qiannan Zhu, Xiaofei Zhou, Jia Wu, Jianlong Tan, and Li Guo. 2019. Neighborhood-aware attentional representation for multilingual knowledge graphs. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI2019)*, pages 1943–1949.