Cross-lingual Knowledge Graph Alignment via Graph Matching Neural Network

Kun Xu[1], Liwei Wang[1], Mo Yu[2], Yansong Feng[3], Yan Song[1], Zhiguo Wang[4], Dong Yu[1]
[1]Tencent AI Lab
[2]IBM T.J. Watson Research
[3]Peking University
[4]Amazon AWS
{syxu828,wlwsjtu1989,zgw.tomorrow}@gmail.com
yum@us.ibm.com, fengyansong@pku.edu.cn, {clksong,dyu}@tencent.com

Abstract

Previous cross-lingual knowledge graph (KG) alignment studies rely on entity embeddings derived only from monolingual KG structural information, which may fail at matching entities that have different facts in two KGs. In this paper, we introduce the *topic entity graph*, a local sub-graph of an entity, to represent entities with their contextual information in KG. From this view, the KB-alignment task can be formulated as a graph matching problem; and we further propose a graph-attention based solution, which first matches all entities in two topic entity graphs, and then jointly model the local matching information to derive a graph-level matching vector. Experiments show that our model outperforms previous state-of-the-art methods by a large margin.

1 Introduction

Multilingual knowledge graphs (KGs), such as DBpedia (Auer et al., 2007) and Yago (Suchanek et al., 2007), represent human knowledge in the structured format and have been successfully used in many natural language processing applications. These KGs encode rich monolingual knowledge but lack the cross-lingual links to bridge the language gap. Therefore, the cross-lingual KG alignment task, which automatically matches entities in a multilingual KG, is proposed to address this problem.

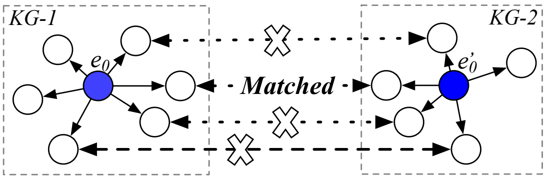

Figure 1: A challenging entity matching example.

Most recently, several entity matching based approaches (Hao et al., 2016; Chen et al., 2016; Sun et al., 2017; Wang et al., 2018) have been proposed for this task. Generally, these approaches first project entities of each KG into low-dimensional vector spaces by encoding monolingual KG facts, and then learn a similarity score function to match entities based on their vector representations. However, since some entities in different languages may have different KG facts, the information encoded in entity embeddings may be diverse across languages, making it difficult for these approaches to match these entities. Figure 1 illustrates such an example where we aim to align e_0 with e'_0, but there is only one aligned neighbor in their surrounding neighbors. In addition, these methods do not encode the entity surface form into the entity embedding, also making it difficult to match entities that have few neighbors in the KG that lacks sufficient structural information.

To address these drawbacks, we propose a *topic entity graph* to represent the KG context information of an entity. Unlike previous methods that utilize entity embeddings to match entities, we formulate this task as a graph matching problem between the topic entity graphs. To achieve this, we propose a novel graph matching method to estimate the similarity of two graphs. Specifically, we first utilize a graph convolutional neural network (GCN) (Kipf and Welling, 2016; Hamilton et al., 2017) to encode two graphs, say G_1 and G_2, resulting in a list of entity embeddings for each graph. Then, we compare each entity in G_1 (or G_2) against all entities in G_2 (or G_1) by using an attentive-matching method, which generates cross-lingual KG-aware matching vectors for all entities in G_1 and G_2. Consequently, we apply another GCN to propagate the local matching information throughout the entire graph. This produces a global matching vector for each topic graph that is used for the final prediction. The

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 3156–3161
Florence, Italy, July 28 - August 2, 2019. ©2019 Association for Computational Linguistics

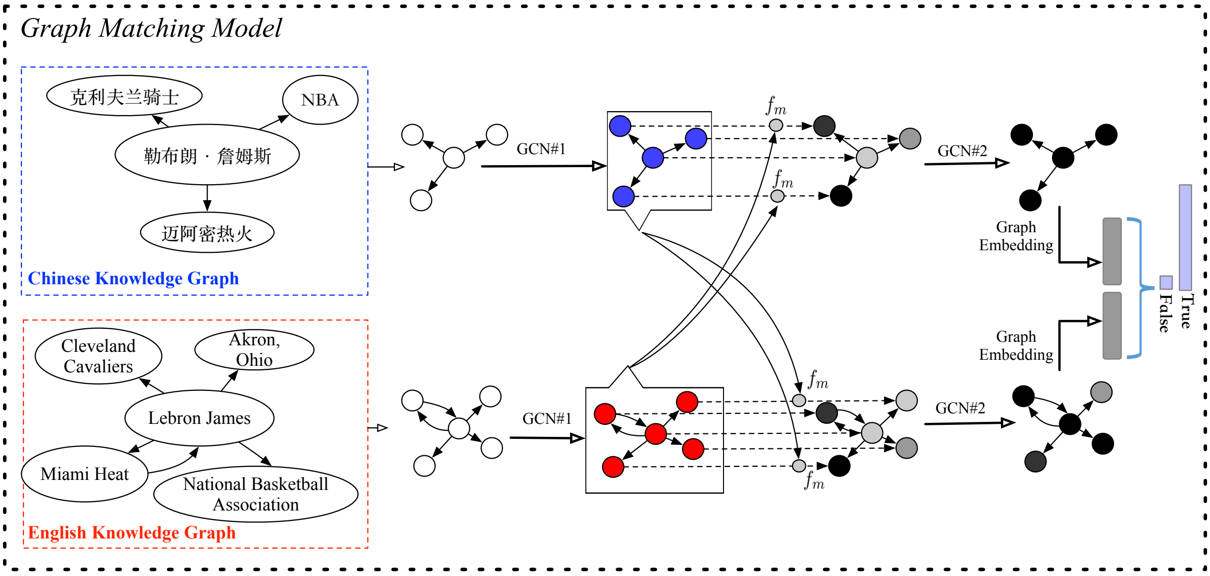

Figure 2: A running example of our model for aligning *Lebron James* in the English and Chinese knowledge graph.

motivation behind is that, the graph convolution could jointly encode all entity similarities, including both the topic entity and its neighbor entities, into a matching vector. Experimental results show that our model outperforms previous state-of-the-art models by a large margin. Our code and data is available at `https://github.com/syxu828/Crosslingula-KG-Matching`.

2 Topic Entity Graph

As indicated in Wang et al. (2018), the local contextual information of an entity in the KG is important to the KG alignment task. In our model, we propose a structure, namely *topic entity graph*, to represent relations among the given entity (called *topic entity*) and its neighbors in the knowledge base. Figure 2 shows the topic graphs of *Lebron James* in the English and Chinese knowledge graph. In order to build the topic graph, we first collect 1-hop neighbor entities of the topic entity, resulting in a set of entities, $\{e_1, ..., e_n\}$, which are the nodes of the graph. Then, for each entity pair (e_i, e_j), we add one directed edge between their corresponding nodes in the topic graph if e_i and e_j are directly connected through a relation, say r, in the KG. Notice that, we do not label this edge with r that e_i and e_j hold in the KG, but just retain r's direction. In practice, we find this strategy significantly improves both the efficiency and performance, which we will discuss in §4.

3 Graph Matching Model

Figure 2 gives an overview of our method for aligning *Lebron James* in the English and Chinese knowledge graph[1]. Specifically, we fist retrieve topic entity graphs of *Lebron James* from two KGs, namely G_1 and G_2. Then, we propose a graph matching model to estimate the probability that G_1 and G_2 are describing the same entity. In particular, the matching model includes the following four layers:

Input Representation Layer The goal of this layer is to learn embeddings for entities that occurred in topic entity graphs by using a GCN (henceforth *GCN*$_1$) (Xu et al., 2018a). Recently, GCN has been successfully applied in many NLP tasks, such as semantic parsing (Xu et al., 2018b), text representation (Zhang et al., 2018), relation extraction (Song et al., 2018) and text generation (Xu et al., 2018c). We use the following embedding generation of entity v as an example to explain the GCN algorithm:

(1) We first employ a word-based LSTM to transform v's entity name to its initial feature vector $\mathbf{a}_v$;

(2) We categorize the neighbors of v into incoming neighbors $\mathcal{N}_{\vdash}(v)$ and outgoing neighbors $\mathcal{N}_{\dashv}(v)$ according to the edge direction.

(3) We leverage an aggregator to aggregate the **incoming** representations of v's incoming neighbors $\{\mathbf{h}^{k-1}_{u\vdash}, \forall u \in \mathcal{N}_{\vdash}(v)\}$ into a single vector, $\mathbf{h}^{k}_{\mathcal{N}_{\vdash}(v)}$, where k is the iteration index. This aggregator

[1] Lebron James is translated to 勒布朗 · 詹姆斯 in Chinese.

feeds each neighbor's vector to a fully-connected neural network and applies an element-wise mean-pooling operation to capture different aspects of the neighbor set.

(4) We concatenate v's current **incoming** representation $\mathbf{h}^{k-1}_{v\vdash}$ with the newly generated neighborhood vector $\mathbf{h}^{k}_{\mathcal{N}_{\vdash}(v)}$ and feed the concatenated vector into a fully-connected layer to update the **incoming** representation of v, $\mathbf{h}^{k}_{v\vdash}$ for the next iteration;

(5) We update the **outgoing** representation of v, $\mathbf{h}^{k}_{v\dashv}$ using the similar procedure as introduced in step (3) and (4) except that operating on the outgoing representations;

(6) We repeat steps (3)∼(5) by K times and treat the concatenation of final incoming and outgoing representations as the final representation of v. The outputs of this layer are two sets of entity embeddings $\{\boldsymbol{e}^1_1, ..., \boldsymbol{e}^1_{|G_1|}\}$ and $\{\boldsymbol{e}^2_1, ..., \boldsymbol{e}^2_{|G_2|}\}$.

Node-Level (Local) Matching Layer In this layer, we compare each entity embedding of one topic entity graph against all entity embeddings of the other graph in both ways (from G_1 to G_2 and from G_2 to G_1), as shown in Figure 2. We propose an attentive-matching method similar to (Wang et al., 2017). Specifically, we first calculate the cosine similarities of entity e^1_i in G_1 with all entities $\{e^2_j\}$ in G_2 in their representation space.

$$\alpha_{i,j} = cosine(\boldsymbol{e}^1_i, \boldsymbol{e}^2_j) \qquad j \in \{1, ..., |G_2|\}$$

Then, we take these similarities as the weights to calculate an attentive vector for the entire graph G_2 by weighted summing all the entity embeddings of G_2.

$$\bar{\boldsymbol{e}}^1_i = \frac{\sum_{j=1}^{|G_2|} \alpha_{i,j} \cdot \boldsymbol{e}^2_j}{\sum_{j=1}^{|G_2|} \alpha_{i,j}}$$

We calculate matching vectors for all entities in both G_1 and G_2 by using a multi-perspective cosine matching function f_m at each matching step (See Appendix A for more details):

$$\boldsymbol{m}^{att}_i = f_m(\boldsymbol{e}^1_i, \bar{\boldsymbol{e}}^1_i)$$
$$\boldsymbol{m}^{att}_j = f_m(\boldsymbol{e}^2_j, \bar{\boldsymbol{e}}^2_j)$$

Graph-Level (Global) Matching Layer Intuitively, the above matching vectors ($\boldsymbol{m}^{att}$s) capture how each entity in G_1 (G_2) can be matched by the topic graph in the other language. However, they are ***local*** matching states and are not sufficient to measure the ***global*** graph similarity. For example, many entities only have few neighbor entities that co-occurr in G_1 and G_2. For those entities, a model that exploits local matching information may have a high probability to incorrectly predict these two graphs are describing different topic entities since most entities in G_1 and G_2 are not close in their embedding space.

To overcome this issue, we apply another GCN (henceforth *GCN*$_2$) to propagate the local matching information throughout the graph. Intuitively, if each node is represented as its own matching state, by design a GCN over the graph (with a sufficient number of hops) is able to encode the global matching state between the pairs of whole graphs. We then feed these matching representations to a fully-connected neural network and apply the element-wise *max* and *mean* pooling method to generate a fixed-length graph matching representation.

Prediction Layer We use a two-layer feed-forward neural network to consume the fixed-length graph matching representation and apply the *softmax* function in the output layer.

Training and Inference To train the model, we randomly construct 20 negative examples for each positive example $<e^1_i, e^2_j>$ using a heuristic method. That is, we first generate rough entity embeddings for G_1 and G_2 by summing over the pretrained embeddings of words within each entity's surface form; then, we select 10 closest entities to e^1_i (or e^2_j) in the rough embedding space to construct negative pairs with e^2_j (or e^1_i). During testing, given an entity in G_1, we rank all entities in G_2 by the descending order of matching probabilities that estimated by our model.

4 Experiments

We evaluate our model on the DBP15K datasets, which were built by Sun et al. (2017). The datasets were generated by linking entities in the Chinese, Japanese and French versions of DBpedia into English version. Each dataset contains 15,000 inter-language links connecting equivalent entities in two KGs of different languages. We use the same train/test split as previous works. We use the Adam optimizer (Kingma and Ba, 2014) to update parameters with mini-batch size 32. The learning rate is set to 0.001. The hop size K of *GCN*$_1$ and *GCN*$_2$ are set to 2 and 3, respectively. The

Method	*ZH-EN*		*EN-ZH*		*JA-EN*		*EN-JA*		*FR-EN*		*EN-FR*	
	@1	@10	@1	@10	@1	@10	@1	@10	@1	@10	@1	@10
Hao (2016)	21.27	42.77	19.52	39.36	18.92	39.97	17.80	38.44	15.38	38.84	14.61	37.25
Chen (2016)	30.83	61.41	24.78	52.42	27.86	57.45	23.72	49.92	24.41	55.55	21.26	50.60
Sun (2017)	41.18	74.46	40.15	71.05	36.25	68.50	38.37	67.27	32.39	66.68	32.97	65.91
Wang (2018)	41.25	74.38	36.49	69.94	39.91	74.46	38.42	71.81	37.29	74.49	36.77	73.06
BASELINE	59.64	72.30	57.66	70.44	67.01	79.53	62.48	77.54	83.45	91.56	81.03	90.79
NodeMatching	62.03	75.12	60.17	72.67	69.82	80.19	66.74	80.10	84.71	92.35	84.15	91.76
Ours												
Hop$_{GCN_2}$ = 1	66.91	77.52	64.01	78.12	72.63	85.09	69.76	83.48	87.62	94.19	87.65	93.66
Hop$_{GCN_2}$ = 3	**67.93**	**78.48**	**65.28**	**79.64**	**73.97**	**87.15**	**71.29**	**84.63**	**89.38**	**95.24**	**88.18**	**94.75**
Hop$_{GCN_2}$ = 5	67.92	78.36	65.21	79.48	73.52	86.87	70.18	84.29	88.96	94.28	88.01	94.37

Table 1: Evaluation results on the datasets.

non-linearity function σ is ReLU (Glorot et al., 2011) and the parameters of aggregators are randomly initialized. Since KGs are represented in different languages, we first retrieve monolingual fastText embeddings (Bojanowski et al., 2017) for each language, and apply the method proposed in Conneau et al. (2017) to align these word embeddings into a same vector space, namely, cross-lingual word embeddings. We use these embeddings to initialize word representations in the first layer of GCN_1.

Results and Discussion. Following previous works, we used *Hits*@1 and *Hits*@10 to evaluate our model, where *Hits*@k measures the proportion of correctly aligned entities ranked in the top k. We implemented a baseline (referred as BASELINE in Table 1) that selects k closest G_2 entities to a given G_1 entity in the cross-lingual embedding space, where an entity embedding is the sum of embeddings of words within its surface form. We also report results of an ablation of our model (referred as *NodeMatching* in Table 1) that uses GCN_1 to derive the two topic entity embeddings and then directly feeds them to the prediction layer without using matching layer. Table 1 summarizes the results of our model and existing works.

We can see that even without considering any KG structural information, the BASELINE significantly outperforms previous works that mainly learn entity embeddings from the KG structure, indicating that the surface form is an important feature for the KG alignment task. Also, the *NodeMatching*, which additionally encodes the KG structural information into entity embeddings using GCN_1, achieves better performance compared to the BASELINE. In addition, we find the graph matching method significantly outperforms all baselines, which suggests that the global context information of topic entities is important to establish their similarities.

Let us first look at the impacts of hop size of GCN_2 to our model. From Table 1, we can see that our model could benefit from increasing the hop size of GCN_2 until it reaches a threshold λ. In experiments, we find the model achieves the best performance when $\lambda = 3$. To better understand on which type of entities that our model could better deal with due to introducing the graph matching layer, we analyze the entities that our model correctly predicts while *NodeMatching* does not. We find the graph matching layer enhances the ability of our model in handling the entities whose most neighbors in two KGs are different. For such entities, although most local matching information indicate that these two entities are *irrelevant*, the graph matching layer could alleviate this by propagating the most relevant local matching information throughout the graph.

Recall that our proposed topic entity graph only retains the relation direction while neglecting the relation label. In experiments, we find incorporating relation labels as distinct nodes that connecting entity nodes into the topic graph hurts not only the performance but efficiency of our model. We think this is due to that (1) relation labels are represented as abstract symbols in the datasets, which provides quite limited knowledge about the relations, making it difficult for the model to learn their alignments in two KGs; (2) incorporating relation labels may significantly increase the topic entity graph size, which requires bigger hop size and running time.

5 Conclusions

Previous cross-lingual knowledge graph alignment methods mainly rely on entity embeddings that derived from the monolingual KG structural information, thereby may fail at matching entities that have different facts in two KGs. To address this, we introduce the topic entity graph to represent the contextual information of an entity within the KG and view this task as a graph matching problem. For this purpose, we further propose a graph matching model which induces a graph matching vector by jointly encoding the entity-wise matching information. Experimental results on the benchmark datasets show that our model significantly outperforms existing baselines. In the future, we will explore more applications of the proposed idea of attentive graph matching. For example, the metric learning based few-shot knowledge base completion (Xiong et al., 2018) can be directly formulated as a similar graph matching problem in this paper.

References

Sren Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. 2007. Dbpedia: A nucleus for a web of open data. In *ISWC/ASWC*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Muhao Chen, Yingtao Tian, Mohan Yang, and Carlo Zaniolo. 2016. Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. *arXiv preprint arXiv:1611.03954*.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, pages 315–323.

William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *arXiv preprint arXiv:1706.02216*.

Yanchao Hao, Yuanzhe Zhang, Shizhu He, Kang Liu, and Jun Zhao. 2016. A joint embedding method for entity alignment of knowledge bases. In *China Conference on Knowledge Graph and Semantic Computing*, pages 3–14. Springer.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. N-ary relation extraction using graph state lstm. *arXiv preprint arXiv:1808.09101*.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *WWW*.

Zequn Sun, Wei Hu, and Chengkai Li. 2017. Cross-lingual entity alignment via joint attribute-preserving embedding. In *International Semantic Web Conference*, pages 628–644. Springer.

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*, pages 349–357.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. *arXiv preprint arXiv:1702.03814*.

Wenhan Xiong, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. 2018. One-shot relational learning for knowledge graphs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1980–1990.

Kun Xu, Lingfei Wu, Zhiguo Wang, and Vadim Sheinin. 2018a. Graph2seq: Graph to sequence learning with attention-based neural networks. *arXiv preprint arXiv:1804.00823*.

Kun Xu, Lingfei Wu, Zhiguo Wang, Mo Yu, Liwei Chen, and Vadim Sheinin. 2018b. Exploiting rich syntactic information for semantic parsing with graph-to-sequence model. *arXiv preprint arXiv:1808.07624*.

Kun Xu, Lingfei Wu, Zhiguo Wang, Mo Yu, Liwei Chen, and Vadim Sheinin. 2018c. Sql-to-text generation with graph-to-sequence model. *arXiv preprint arXiv:1809.05255*.

Yue Zhang, Qi Liu, and Linfeng Song. 2018. Sentence-state lstm for text representation. *arXiv preprint arXiv:1805.02474*.

A Matching Function f_m

f_m is a multi-perspective cosine matching function that compares two vectors

$$\boldsymbol{m} = f_m(\boldsymbol{v}_1, \boldsymbol{v}_2; \boldsymbol{W})$$

where $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are two d-dimensional vectors, $\boldsymbol{W} \in \Re^{l \times d}$ is a trainable parameter with the shape $l \times d$, l is the number of perspectives, and the returned value $\boldsymbol{m}$ is a l-dimensional vector $\boldsymbol{m} =$

$[m_1, ..., m_k, ..., m_l]$. Each element $m_k \in \boldsymbol{m}$ is a matching value from the k-th perspective, and it is calculated by the cosine similarity between two weighted vectors

$$m_k = cosine(W_k \circ \boldsymbol{v}_1, W_k \circ \boldsymbol{v}_2)$$

where $\circ$ is the element-wise multiplication, and W_k is the k-th row of $\boldsymbol{W}$, which controls the k-th perspective and assigns different weights to different dimensions of the d-dimensional space.