Dynamic Knowledge Graph Alignment

Yuchen Yan, Lihui Liu, Yikun Ban, Baoyu Jing, Hanghang Tong
University of Illinois at Urbana-Champaign, Urbana, IL, USA
{yucheny5, lihuil2, yikunb5, baoyuj, htong}@illinois.edu

Abstract

Knowledge graph (KG for short) alignment aims at building a complete KG by linking the shared entities across complementary KGs. Existing approaches assume that KGs are static, despite the fact that almost every KG evolves over time. In this paper, we introduce the task of dynamic knowledge graph alignment, the main challenge of which is how to efficiently update entity embeddings for the evolving graph topology. Our key insight is to view the parameter matrix of GCN as a feature transformation operator and decouple the *transformation* process from the *aggregation* process. Based on that, we first propose a novel base algorithm (DINGAL-B) with topology-invariant mask gate and highway gate, which consistently outperforms 14 existing knowledge graph alignment methods in the static setting. More importantly, it naturally leads to two effective and efficient algorithms to align dynamic knowledge graph, including (1) DINGAL-O which leverages previous parameter matrices to update the embeddings of affected entities; and (2) DINGAL-U which resorts to newly obtained anchor links to fine-tune parameter matrices. Compared with their static counterpart (DINGAL-B), DINGAL-U and DINGAL-O are $10\times$ and $100\times$ faster respectively, with little alignment accuracy loss.

Introduction

Knowledge graph (KG) is an integral component in a multitude of applications, ranging from dialogue systems (Xu, Bao, and Zhang 2020), question-answering (Yu et al. 2017) to search engine optimization (Xiong, Power, and Callan 2017). A variety of KGs have been constructed in recent years, such as YAGO (Rebele et al. 2016) and DBpedia (Lehmann et al. 2015).

Data in KGs is organized in the standard `(head, relation, tail)` triplet form. Different KGs usually collect different aspects of the knowledge, and they are complementary to each other (Wang et al. 2018). As illustrated in Figure 1, one KG (green) records the information of Barack Obama from the political aspect, whereas the other (blue) is mainly about his personal relations. However, the two KGs cannot be easily merged together due to the discrepancy of entity mentions, e.g., Obama v.s. Barack Obama. The task of linking the entities sharing the same identity across KGs is referred to as the knowledge graph alignment.

Copyright © 2021, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

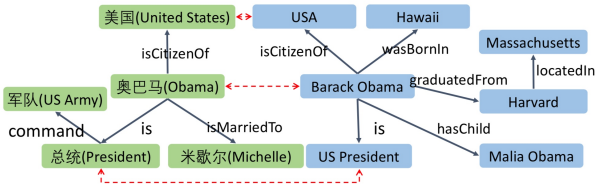

Figure 1: Knowledge graph alignment.

Earlier approaches for knowledge graph alignment date back to symbolic based methods (Suchanek, Abiteboul, and Senellart 2011), where manually built features or rules are the keys to align entities (Mahdisoltani, Biega, and Suchanek 2014). As knowledge graph embedding becomes popular (Bordes et al. 2013), many embedding based knowledge graph alignment methods are proposed (Sun, Hu, and Li 2017; Chen et al. 2016, 2018). With the advent of Graph Convolutional Network (GCN) (Kipf and Welling 2016) and Graph Neural Network (GNN) (Hamilton, Ying, and Leskovec 2017), more recent works (Wang et al. 2018; Cao et al. 2019a; Wu et al. 2019a; Sun et al. 2019) often rely on GCN or GNN to capture the heterogeneity of KGs. The state-of-the-art performance of embedding based knowledge graph alignment methods can be credited to their ability to take advantage of various features in the KGs, such as node attribute (Sun, Hu, and Li 2017), relation information (Wu et al. 2019a) and rule inference (Cao et al. 2019a).

Most, if not all, existing knowledge graph alignment approaches assume that the input knowledge graphs are static, despite the fact that almost every knowledge graph evolves over time. For example, `(Donald Trump, homeLocated, White House)` along with many other relevant triples were added into YAGO in 2017.

Simply retraining the alignment model from scratch is computationally costly, and might even be infeasible for real-time applications such as knowledge based question-answering system (Savenkov and Agichtein 2016).

In this paper, we first formally introduce the problem of dynamic knowledge graph alignment, and then we tackle this problem based on GCN. In the dynamic setting, a major hurdle lies in the difficulty of updating the embeddings, which is in turn rooted in the coupling between the graph topology,

the embeddings (or the features) [1] and the parameter matrix. To address this issue, the key insight is to view the parameter matrix of GCN as a feature transformation operator, and thus distance or weaken the coupling between the *transformation* process and the *aggregation* process. Armed with this idea, we propose a novel algorithm (DINGAL-B) based on gated GCN, including a mask gate and a highway gate. The empirical evaluations demonstrate that DINGAL-B consistently outperforms 14 existing knowledge graph alignment methods in the static setting. More importantly, both mask gate and highway gate in DINGAL-B are intentionally designed to be decoupled from the underlying graph topology. This brings a crucial benefit in the dynamic setting, and it naturally leads to two effective and efficient algorithms to align dynamic knowledge graph, including (1) DINGAL-O (Old-parameter) which leverages previous parameter matrices to update the embeddings of affected entities; and (2) DINGAL-U (Updated-parameter) which resorts to newly obtained anchor links to fine-tune parameter matrices.

The main contributions of the paper are summarized as follows.

- **Problem Definition**. To the best of our knowledge, we are the first to study the dynamic knowledge graph alignment problem.
- **New Algorithms.** We propose a family of knowledge graph alignment algorithms (DINGAL), including DINGAL-B for static alignment, and DINGAL-O and DINGAL-U for dynamic alignment.
- **Empirical Evaluations.** Extensive evaluations are conducted on the benchmark $DBP15K$ (Sun, Hu, and Li 2017) datasets. In the static setting, the proposed DINGAL-B model consistently outperforms 14 state-of-the-art methods. In the dynamic setting, the proposed DINGAL-O and DINGAL-U are (1) $10\times$ faster *and* better than the existing static alignment methods; and (2) $10\times$ to $100\times$ faster than their static counterpart (DINGAL-B) with little alignment accuracy loss.

Preliminary

Knowledge Graph Alignment. A knowledge graph is represented as $G = (E, R, T)$, where E, R, and T are the sets of entities, relations and triples respectively. Let $G_1 = (E_1, R_1, T_1)$ and $G_2 = (E_2, R_2, T_2)$ be two KGs to be aligned. If an entity e_1 in G_1 corresponds to another entity e_2 in G_2 (e.g., entities linked by the red dashed arrows in Figure 1), we call (e_1, e_2) as an *alignment pair*. If an alignment pair (e_1, e_2) is a known prior, we refer to it as an *anchor link*. Given a limited number of anchor links $\mathcal{L} = \{(e_{i_1}, e_{i_2}) | e_{i_1} \in E_1, e_{i_2} \in E_2\}$, the task of knowledge graph alignment is to find all alignment pairs across two KGs.

Dynamic Knowledge Graph Alignment. Real-world KGs are evolving over time, e.g., the arrival of the new entities and the vanishing of the out-dated relations. It is desirable to not only find the alignment for the newly arrived entities, but also leverage the new information to refine the alignment for the existing entities. For dynamic knowledge graph alignment task, there might exist some anchor links among two newly arrived node sets $\tilde{E}_1$ and $\tilde{E}_2$ in two knowledge graphs. The task of dynamic knowledge graph alignment is to find all alignment pairs for newly arrived node sets and update the alignment for the existing entities, without rerunning or retraining the static knowledge graph alignment algorithm.

[1] We use the term embedding and feature interchangeably in this paper.

Graph Convolutional Network. The proposed family of algorithms (DINGAL) is built upon GCN (Kipf and Welling 2016). Given an adjacency matrix $A \in \mathbb{R}^{n \times n}$, where n is the number of nodes, the l-th layer of GCN is defined as:

$$X^{l+1} = \sigma(L X^l W^l) \tag{1}$$

where $L = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$, $\tilde{A} = A + I$, and $\tilde{D}$ is the degree matrix of $\tilde{A}$, X^l and X^{l+1} denote the node embeddings after the $(l-1)$-th and l-th layer respectively, and σ denotes the activation function.

Method

Overview

As shown in Figure 2, the proposed DINGAL family has three models, including DINGAL-B which is a base model for static knowledge graph alignment, DINGAL-O and DINGAL-U for dynamic knowledge graph alignment. The key difference between DINGAL-O and DINGAL-U is that DINGAL-O extracts the embeddings of the *affected* nodes by leveraging the pretrained parameters of DINGAL-B, while DINGAL-U leverages the new anchor links to update the parameters, which are in turn used to update the embeddings of all the entities. DINGAL is built upon the GCN in Equation (1). As illustrated in Figure 3, the classic viewpoint of GCN is an *aggregation-then-transformation* function. Specifically, the nodes first *aggregate* the features of their neighbors via the normalized Laplacian L, and then the aggregated features are projected into a hidden space by a linear transformation W^l (W_g^l in Figure 3).

With this viewpoint, it is very challenging to update the embeddings in the dynamic setting, since any change in the graph (reflected in the Laplacian L) will impact both aggregation and *transformation* operations in GCN. Model effective and efficient for dynamic alignment task should meet the requirement that the model's parameters are robust to graph topology's change. In a better case, the dynamic process will just affect a small part of node embeddings.

To address this challenge, the key idea is to disentangle the graph topology from the parameter matrix of GCN, and design a topology-invariant model to extract node embeddings. We first view GCN as a *transformation-before-aggregation* function by changing the orders of matrix multiplications. We first project the node embeddings matrix X^l to a hidden space via the linear transformation W^l, and then aggregate the projected features of the neighboring nodes based on L. In this way, we weaken the coupling between the graph topology L and the parameter matrix W^l. Based on this viewpoint, we introduce topology-invariant functions, including

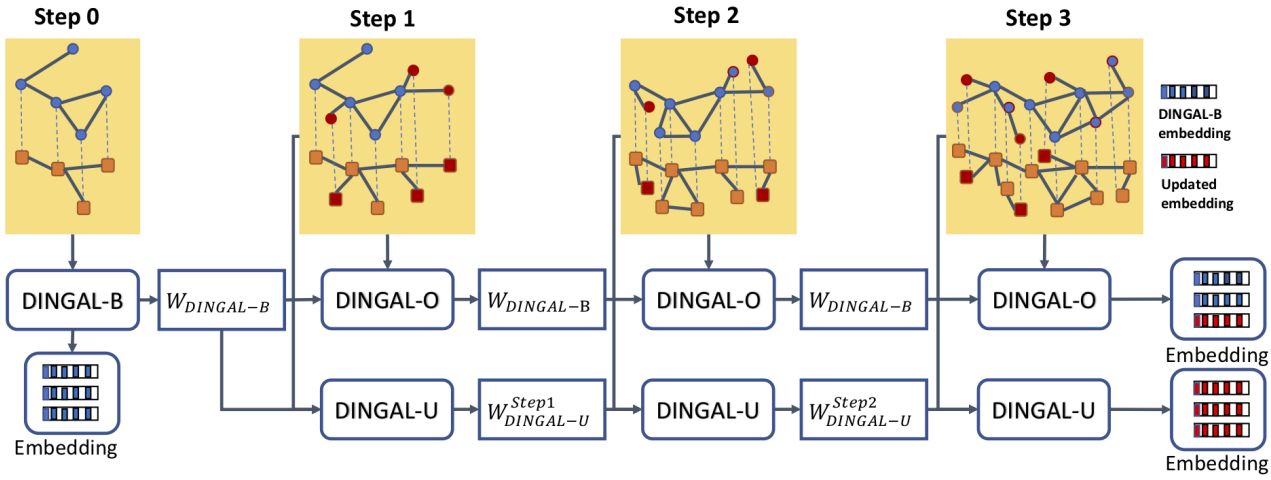

Figure 2: Overview of DINGAL algorithm family.

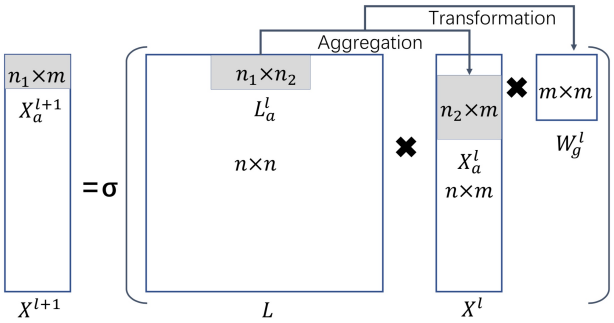

Figure 3: The structures of GCN layer for retraining DINGAL-B(transparent part) and adopting DINGAL-O(shaded part).

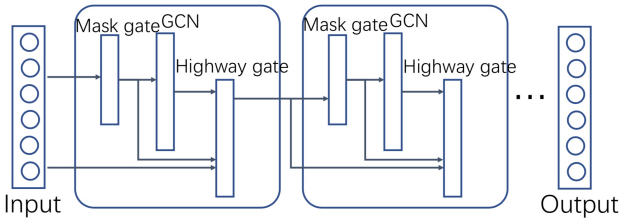

Figure 4: The structure of DINGAL-B.

a mask gate and a highway gate (Wu et al. 2019a; Srivastava, Greff, and Schmidhuber 2015), which are intentionally decoupled with the graph topology by bypassing graph laplacian. The target to meet dynamic task requirement determines the design strategy of DINGAL-B and DINGAL-B will lead to two effective and efficient algorithms (DINGAL-O and DINGAL-U) in the dynamic setting.

DINGAL-B

We first present a base version DINGAL-B of the DINGAL family on the static knowledge graphs. Given the aforementioned intuition, we seek to build topology-invariant functions based upon the *transformation* process of the GCN and avoid involving the *aggregation* process. In this way, the evolution of graph topology will only incur limited influence on such functions.

We view the two KGs as one combined graph with adjacency matrix A and feature matrix X^0, where A is the adjacency matrices of the two KGs: A_1 and A_2. Then, we propose two topology-invariant functions, i.e., a mask gate and a highway gate, as shown in Figure 4. In the l-th layer of DINGAL-B, for any entity e_i with input feature X_i^l, it will first enter a topology-invariant mask gate M^l:

$$M^l(X_i^l) = X_i^l \odot \sigma(W_m^l) \tag{2}$$

$\odot$ denotes Hadamard product and it determines relative importance of different dimensions of the features, which is similar to the attention mechanism.

Then, $M^l(X^l)$ will be fed into a GCN layer as follows:

$$X^{l+1} = \sigma(LM^l(X^l)W_g^l) \tag{3}$$

We use $L = \tilde{D}^{-1}\tilde{A}$ rather than $L = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ in the original GCN for computational efficiency. Detailed explanations are presented in the Appendix.

Since the GCN layer has a feature transformation operation on the original distribution of feature, we propose to use a highway gate similar to (Wu et al. 2019a; Sun et al. 2019), an interpolation between the original feature distribution and the transformed feature distribution, which is disentangled with the topology. Given the masked input $M^l(X^l)$, the proposed highway gate is:

$$T^l(X^l) = \sigma(M^l(X^l)W_h^l) \tag{4}$$

The final output of one DINGAL-B layer is:

$$X^{l+1} = T^l(X^l) \odot X^{l+1} + (1 - T^l(X^l)) \odot X^l \tag{5}$$

With the final embeddings matrix X of two KGs, we use Manhattan distance to measure the distance between two nodes:

$$d(e_{i_1}, e_{i_2}) = |X_{e_{i_1}} - X_{e_{i_2}}| \tag{6}$$

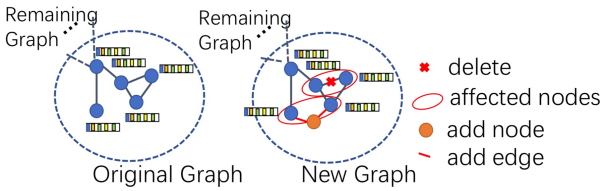

Figure 5: The affected entities in this dynamic process.

We use the same margin ranking loss function as previous works:

$$J = \sum_{(e_{i_1}, e_{i_2})} \sum_{(e'_{i_1}, e'_{i_2})} \max\{0, d(e_{i_1}, e_{i_2}) - d(e'_{i_1}, e'_{i_2}) + \gamma\} \quad (7)$$

where $\gamma > 0$ is a margin hyper-parameter, the anchor link $(e_{i_1}, e_{i_2}) \in \mathcal{L}$ is the training set and $(e'_{i_1}, e'_{i_2}) \in \mathcal{L}'$ is the negative sampling set. For a given (e_{i_1}, e_{i_2}), we construct the negative samples by replacing e_{i_1} with e'_{i_1} or replacing e_{i_2} with e'_{i_2} such that e'_{i_1} and e'_{i_2}, respectively, have the smallest distances d to e_{i_2} and e_{i_1} in the embedding space.

DINGAL-O

For DINGAL-O, we keep all the parameters of DINGAL-B and update the embeddings of those affected entities in the dynamic process. We define one-hop affected entities as the new entities and the old entities having changed (deleted/added) edges. Note that the deleted entities are not taken into consideration because they are not involved in the dynamic alignment task. The definition of the affected entity can be extended to multi-hop. Figure 5 presents an example of an affected local part of the KG, where the number of the affected entities n_1 is 5 and the size of one-hop neighborhood of these affected entities n_2 is 6. Instead of updating the global graph topology and rerunning the entire model for the KG, we focus on the affected local regions and propose a novel local updating strategy on DINGAL-B and we refer to the new model as DINGAL-O.Since all the components of DINGAL-B are specially designed to be decoupled with the graph topology, DINGAL-O can utilize DINGAL-B's parameters and update a small part affected nodes' embeddings with a little accuracy loss efficiently. As shown in Figure 3, for the GCN layer within the l-th layer of DINGAL-O, we construct the affected local Laplacian matrix $L^l_a \in \mathbb{R}^{n_1 \times n_2}$ upon the global Laplacian matrix $L \in \mathbb{R}^{n \times n}$, where n is the total number of nodes (including the new nodes), $n_1, n_2 \ll n$ are the number of the affected nodes and the size of the affected one-hop neighborhood. Based on the L^l_a, the node embedding X^{l+1}_a for the affected nodes can be easily obtained by the embedding X^l_a of the local neighborhood:

$$X^{l+1}_a = \sigma(L^l_a M^l(X^l_a) W^l_g) \quad (8)$$

Compared with retraining the entire GCN layer, DINGAL-O significantly saves both space and time cost.

DINGAL-U

DINGAL-O extracts the node embeddings for the affected nodes by directly leveraging the pre-trained parameters of DINGAL-B without updating them. In this section, we take one step further and consider the setting that the anchor links are provided in the new entities. We propose a novel DINGAL-U, which leverages the new anchor links to fine-tune the parameters of DINGAL-B and update the embedding for all entities. To be more specific, DINGAL-U treats the new entities as the affected nodes and adopts the same updating function as DINGAL-O (Equation (8)) in the GCN layer. The loss function for DINGAL-U is the same as Equation (7), where only newly added anchor links will be used as positive samples to fine-tune the parameters. DINGAL-U saves the training time by avoiding re-training the weight matrix from scratch, while DINGAL-O uses the old weight matrix to obtain the embedding of new nodes, and thus DINGAL-O even more efficient than DINGAL-U. Note that DINGAL-U only uses the newly added entities to fine-tune the parameters.

Experiment

We evaluate the proposed DINGAL family in the following two aspects. First, following (Chen et al. 2016; Sun, Hu, and Li 2017), we evaluate DINGAL-B in the standard static knowledge graph alignment setting. Second, for DINGAL-O and DINGAL-U, we evaluate their effectiveness and efficiency by comparing with rerunning static knowledge graph alignment methods in the dynamic setting.

Experimental Setup

Datasets *Datasets for the static setting.* We use $DBP15K$ (Sun, Hu, and Li 2017) benchmark datasets, including $DBP15K_{ZH-EN}$, $DBP15K_{JA-EN}$ and $DBP15K_{FR-EN}$ built on Chinese, English, Japanese and French versions of DBpedia. Each dataset provides two KGs in different languages with 15K pre-aligned entity pairs. the detailed statistics of datasets is shown in Appendix for page limitation.

Datasets for the dynamic setting. We randomly split the 15,000 ground-truth aligned pairs in each of $DBP15K$ datasets into three dynamic time steps (t_0, t_1 and t_2) with a ratio 8:1:1, which is 12,000:1,500:1,500 for each time step. For the start time step t_0, the KG pair removes 3,000 aligned entity pairs in ground-truth set and the edges linked to them. For any entity not belonging to the ground-truth, if it becomes an isolated entity due to the temporal change, it is deleted. At time step t_1, 1,500 aligned pairs together with the isolated entities linked to them will be added to the KG pairs at t_0, which will form the new KG pairs at time step t_1. At time step t_2, by adding the remaining 1,500 aligned pairs into the KG pairs at t_1, we recover the original $DBP15K$ datasets.

Metrics Hit@k is the proportion of correctly aligned entities ranked in the top-k list. We strictly follow the evaluation protocol used in the entity linking and KG alignment community(e.g., GCN-Align, RDGCN, HGCN-JE and AliNet etc.). When calculating Hit@k for one specific entity from KG1 in the test set, it *only* considers entities in test set from KG2 as candidate set instead of all KG2 entities. This evaluation method is fair to compare different methods for a *given* test set.

Models	ZH-EN		JA-EN		FR-EN	
	Hits@1	Hits@10	Hits@1	Hits@10	Hits@1	Hits@10
MtransE (Chen et al. 2016)	30.8	61.4	27.9	57.5	24.4	55.6
JAPE (Sun, Hu, and Li 2017)	41.2	74.5	36.3	68.5	32.4	66.7
BootEA (Sun et al. 2018)	62.9	84.8	62.2	85.4	65.3	87.4
GCN-Align (Wang et al. 2018)	41.3	74.4	39.9	74.5	37.3	74.5
SEA (Pei et al. 2019)	42.4	79.6	38.5	78.3	40.0	79.7
RSN (Guo, Sun, and Hu 2019)	50.8	74.5	50.7	73.7	51.6	76.8
MuGNN (Cao et al. 2019a)	49.4	84.4	50.1	85.7	49.5	87.0
RDGCN (Wu et al. 2019a)	70.8	84.6	76.7	89.5	88.6	95.7
GMNN (Xu et al. 2019)	67.9	78.5	74.0	87.2	89.4	95.2
KECG (Li et al. 2019)	47.8	83.5	49.0	84.4	48.6	85.1
HGCN-JE (Wu et al. 2019b)	72.0	85.7	76.6	89.7	89.2	96.1
NAEA (Zhu et al. 2019)	65.0	86.7	64.1	87.3	67.3	89.4
MMEA (Shi and Xiao 2019)	68.1	86.7	65.5	85.9	67.7	89.0
AliNet (Sun et al. 2019)	53.9	82.6	54.9	83.1	55.2	85.2
DINGAL-B(w/o highway gate)	42.7	70.6	43.1	73.3	55.9	82.3
DINGAL-B(w/o mask gate)	69.8	84.8	76.2	88.9	91.0	97.2
DINGAL-B(one-layer)	70.0	84.3	77.0	89.5	91.0	96.9
DINGAL-B	**72.5**	**87.4**	**78.2**	**91.3**	**92.0**	**97.9**

Table 1: The alignment performance for $DBP15K$ with 0.3 training ratio.

Baselines For the static setting, we use 14 recent embedding based state-of-the-art methods as baselines for DINGAL-B, including MTransE (Chen et al. 2016), JAPE (Sun, Hu, and Li 2017), BootEA (Sun et al. 2018), GCN-Align (Wang et al. 2018), MMEA (Shi and Xiao 2019), NAEA (Zhu et al. 2019), KECG (Li et al. 2019), RDGCN (Wu et al. 2019a), MuGNN (Cao et al. 2019a), GMNN (Xu et al. 2019), SEA (Pei et al. 2019), RSN (Guo, Sun, and Hu 2019), HGCN-JE (Wu et al. 2019b), AliNet (Sun et al. 2019). We report the same results as in the corresponding original papers. For the dynamic setting, we use 8 baselines, including MTransE (Chen et al. 2016), BootEA (Sun, Hu, and Li 2017), GCN-Align (Wang et al. 2018), MMEA (Shi and Xiao 2019), KECG (Li et al. 2019), NAEA (Zhu et al. 2019), MuGNN (Cao et al. 2019a) and RDGCN (Wu et al. 2019a). The implementation of the first 6 baseline methods are from EAkit, an open-source entity alignment toolkit.

Implementaion details In the static setting, we use the same split as the previous works, i.e., 30% for the training set and 70% for the test set. The epoch number is set as 1500. The number of negative samples for each positive sample is 125. The learning rate is 0.001. We use a two-layer DINGAL-B in the experiment. The margin hyper-parameter γ in Equation (7) is 1. The embedding dimension is 300. We use Google translation api and $glove.840B.300d$ as initial node features, which is the same as GMNN and RDGCN.

In the dynamic setting, for each time step, we split the ground-truth with 30% for training and 70% for test. The size of the training set is 3600:450:450 and the test set is 8400:1050:1050 for new entities at each time step[2]. The training set at t_1 and t_2 are used by DINGAL-U. DINGAL-O does not need the new training set at t_1 or t_2 because it keeps the same parameter matrices as DINGAL-B from t_0. For baselines and DINGAL-B, we retrain each model from scratch for each step. The learning rate is 0.005 and the epoch number is 700. The sizes of entire training sets and entire test sets for these methods are 3600:4050:4500 and 8400:9450:10500 respectively at each time step. The fine-tuning learning rate for DINGAL-U is 0.001. It has 80 training epochs. The remaining hyper-parameters are the same as the static setting. The experiment are run on a 1080-Ti GPU.

[2]The static setting result in Table 1 is the largest dataset at t_2.

Static Knowledge Graph Alignment

A - Alignment Performance Comparison. From Table 1, we observe that the proposed DINGAL-B consistently outperforms all 14 recent embedding-based methods on both Hit@1 and Hit@10 metrics for all three benchmark datasets.

Among different baselines, GCN-based methods (e.g., RDGCN and HGCN-JE) have a small advantage over the methods in translational family (e.g., JAPE and BootEA) for the reason that GCN offers a better capacity to capture the attribute information. Besides, some methods (such as HGCN-JE, RDGCN, and GMNN) utilizing $glove.840B.300d$ have a better performance with more than 10% improvement in Hit@1 and Hit@10 metrics than other methods, which corroborates the importance of having good initialization feature vectors. DINGAL-B has an improvement in Hit@1 and Hit@10 about 1.8% over HGCN-JE, 2.3% over RDGCN, and 4.5% over GMNN, all of which are the recent strong baselines. The proposed DINGAL-B shares some similar structures as some baseline methods, such as the highway gate used in RDGCN, HGCN-JE, and AliNet. We further analyze the detailed implementation of these methods. We

Models	ZH-EN		JA-EN		FR-EN	
	Hits@1	Hits@10	Hits@1	Hits@10	Hits@1	Hits@10
MtransE (Chen et al. 2016)	33.7	67.1	33.1	65.3	28.7	61.1
BootEA (Sun et al. 2018)	44.1	76.9	44.8	78.5	45.2	80.8
GCN-Align (Wang et al. 2018)	31.8	68.0	34.1	69.9	32.9	69.7
NAEA (Zhu et al. 2019)	17.4	23.2	16.4	23.9	22.2	34.4
MMEA (Shi and Xiao 2019)	22.9	48.8	23.9	50.6	27.4	60.2
MuGNN (Cao et al. 2019a)	38.2	73.7	36.4	74.0	35.5	74.9
KECG (Li et al. 2019)	33.6	69.5	34.9	72.0	36.4	73.2
RDGCN (Wu et al. 2019a)	69.6	84.9	75.7	88.9	87.3	97.1
DINGAL-B	**72.7**	**88.0**	**78.3**	**91.2**	**91.8**	**97.9**
DINGAL-O	71.6	86.8	77.3	90.3	**91.8**	97.6
DINGAL-U	71.7	87.5	77.4	90.6	91.4	97.5

Table 2: The alignment performance for the entire test set at t_1 with 0.3 training ratio.

Models	ZH-EN		JA-EN		FR-EN	
	Hits@1	Hits@10	Hits@1	Hits@10	Hits@1	Hits@10
MtransE (Chen et al. 2016)	61.5	86.8	57.7	82.9	51.4	80.3
BootEA (Sun et al. 2018)	69.5	90.1	71.6	92.4	72.6	94.8
GCN-Align (Wang et al. 2018)	61.2	89.0	62.7	88.3	59.2	89.4
NAEA (Zhu et al. 2019)	32.3	48.7	23.4	45.1	35.2	51.7
MMEA (Shi and Xiao 2019)	45.0	66.5	43.7	67.3	52.3	77.1
MuGNN (Cao et al. 2019a)	64.3	90.9	65.9	90.8	66.6	94.1
KECG (Li et al. 2019)	61.4	89.0	64.4	89.7	63.6	89.4
RDGCN (Wu et al. 2019a)	81.6	91.8	86.0	93.9	93.1	98.1
DINGAL-B	85.6	95.4	**89.4**	**97.3**	**97.1**	**99.9**
DINGAL-O	85.0	95.1	88.4	97.1	96.4	99.4
DINGAL-U	**85.7**	**95.8**	88.8	97.2	96.5	99.4

Table 3: The alignment performance for the newly added test set at t_1 with 0.3 training ratio.

find that when they (RDGCN, HGCN-JE, and AliNet) implement their novel components in the corresponding models, they bring in some unnecessary parameters, which cause the over-fitting problem that leads to the inferior performance.

B - Ablation Study.

We conduct the ablation study for DINGAL-B. We design three DINGAL-B variants, including DINGAL-B without highway gate, DINGAL-B without mask gate, and one layer DINGAL-B, whose performances are shown at the bottom of Table 1. We can see that DINGAL-B without highway gate has a dramatic drop of about 33% in Hit@1 and 16% in Hit@10, which is consistent with ablation study in previous models (Wu et al. 2019a,b). The reason is that GCN's weight matrix primarily acts as a feature transformation, which is in turn determined by the relationship between different dimensions of the node feature instead of the graph topology. The transformed feature space alone performs poorly. The highway gate performs a linear interpolation between the original feature space and transformed feature space, whose interpolation coefficient is obtained via training. Based on the result, the interpolation between the old and transformed spaces is the key that makes the highway gate perform well. DINGAL-B without mask gate and one-layer DINGAL-B both have around a 1.5% drop in Hit@1 and a 2.3% drop in Hit@10, which shows the effectiveness of mask gate component and setting the layer number as 2.

Dynamic Knowledge Graph Alignment

A - Effectiveness. The results of the entire test set and the newly added test set at t_1 [3] are shown in Table 2 and Table 3 respectively. DINGAL-B still achieves the highest performance for the entire test set (9,450 pairs) and the newly added test set (1,050 pairs) after retraining. With the same parameters as DINGAL-B at time step t_0, DINGAL-O updates the embeddings of affected entities in t_1. It only has a small drop in the performance compared to retraining DINGAL-B. By using a small newly added training set, DINGAL-U's performance lies between DINGAL-O and retraining DINGAL-B. Although the performance of DINGAL-O and DINGAL-U is not as high as DINGAL-B, they only have a small (about 1%) drop in Hit@1 metrics. In the meanwhile, both DINGAL-O and DINGAL-U outperform the 8 baselines even though these baselines need to be retrained when KGs change over

[3]The remaining experiment results are attached in the Appendix due to the page limitation.

time. *Quality-speed Trade-off.* The running time and Hit@1 for the entire test set on $DBP15K_{ZH-EN}$ dataset of different methods at t_1 are shown in Figure 6. For $DBP15K$ dataset, DINGAL-U is $10\times$ faster than most static knowledge graph alignment algorithms. Compared with DINGAL-U, DINGAL-O achieves another $10\times$ acceleration, which spends only 3-5s at each time step. Even though some baseline methods (e.g., GCN-Align) have a comparable running time as DINGAL-U, their performance is significantly lower than DINGAL-U and DINGAL-O. Among other baselines, RDGCN has a competitive Hit@1 as DINGAL-U and DINGAL-B, yet with a significantly longer running time.

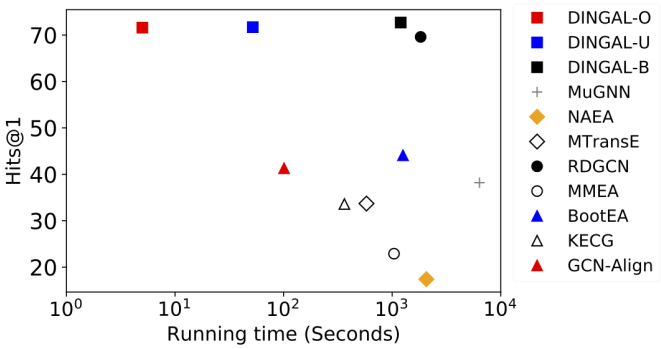

Figure 6: The running time of different methods at t_1.

Related Work

Knowledge Graph Alignment

Knowledge graph alignment, also known as entity linking or entity alignment, first appears in the context of knowledge base integration. In (Rinser, Lange, and Naumann 2013; Nguyen et al. 2011), well-defined schema and ontology are used to boost the entity mapping. Since the emergence of TransE (Bordes et al. 2013), many works seek to solve the alignment problem from the embedding perspective. JE (Hao et al. 2016), MTransE (Chen et al. 2016), IPTransE (Zhu et al. 2017) and BootEA (Sun et al. 2018) are based on translational family embedding methods. SEA (Pei et al. 2019) pays attention to degree-aware embedding and KDCoE (Chen et al. 2018) co-trains the embeddings for entity description and multilingual KGs. To make better use of entity attributes, GNN has been applied to knowledge graph alignment. Among others, (Wang et al. 2018) and (Cao et al. 2019b) are two recent representative knowledge graph alignment works utilizing GNN. In addition, some works (Wu et al. 2019a; Mao et al. 2020; Sun et al. 2019) aim to use relation attributes or distant neighborhood information, which might lead to further performance improvement.

Static Graph Embedding

Static graph embedding (i.e., representation learning) aims to embed the nodes or entities in the graph to a low dimensional space. It can be categorized as network representation learning (NRL) and knowledge graph embedding. For NRL, DeepWalk(Perozzi, Al-Rfou, and Skiena 2014), Node2Vec(Grover and Leskovec 2016) and LINE(Tang et al. 2015) are designed to preserve local context of nodes. In order to further incorporate attributes information, algorithms like (Ou et al. 2015, 2016; Wang et al. 2017b) and graph convolutional network (Bruna et al. 2014; Hamilton, Ying, and Leskovec 2017) have been proposed. Knowledge graph embedding methods can be roughly divided into translational distance models, semantic matching models and others (Wang et al. 2017a). Translational distance models (Bordes et al. 2013) are based on distance score for different entities. Semantic matching models like RESCAL (Nickel, Tresp, and Kriegel 2011) seek to use a matrix to measure the similarity between two entities. SME (Bordes et al. 2014) uses a neural network to conduct this task. Other works include relation paths (Lao and Cohen 2010) and entity types (Nickel, Tresp, and Kriegel 2012).

Dynamic Graph Embedding

Research on dynamic graph embedding can be divided into *continuous* and *discrete* methods. For continuous graph representation learning, the embedding can be thought as a function of the time t. (Dasgupta, Ray, and Talukdar 2018) formulates a vector of time for graphs at different time steps. Time is used as a regularizer in (Chakrabarti, Kumar, and Tomkins 2006). In (Nguyen et al. 2018), the event stream model has been proposed to model continuous dynamic graphs. The dynamic knowledge graph alignment task is more related to discrete graph representation learning because the dynamic change in the alignment task occurs step by step. (Sun et al. 2012) is the first to study relation formation between nodes. (Zhou et al. 2018) views the dynamic graph as a process of triadic closure. (Sharan and Neville 2008) gives a large weight to the most recent graph snapshot when combined with previous ones. (Sajjad, Docherty, and Tyshetskiy 2019) has extended the random walk based method by keeping the valid old random walks in previous time steps and sampling some new walks with new nodes.

Conclusion

In this paper, we propose a family of algorithms (DINGAL) for knowledge graph alignment based on graph convolutional network. The key idea is to distance the coupling between the parameter matrix in GCN and the underlying graph topology. By introducing two topology-invariant functions, the proposed DINGAL-B consistently outperforms 14 existing knowledge graph alignment methods in the static setting. Based on that, we propose two effective and efficient algorithms to align dynamic knowledge graph, including (1) DINGAL-O which leverages previous parameter matrices to update the embeddings of affected entities; and (2) DINGAL-U which resorts to newly obtained anchor links to fine-tune parameter matrices. Compared with their static counterpart (DINGAL-B), DINGAL-U and DINGAL-O are $10\times$ and $100\times$ faster respectively, with little alignment accuracy loss.

Acknowledgements

This work is supported by National Science Foundation under grant No. 1947135, and 2003924 and by the NSF Program on Fairness in AI in collaboration with Amazon under award

No. 1939725. The content of the information in this document does not necessarily reflect the position or the policy of the Government or Amazon, and no official endorsement should be inferred. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

Bordes, A.; Glorot, X.; Weston, J.; and Bengio, Y. 2014. A semantic matching energy function for learning with multi-relational data. *Machine Learning* 94(2): 233–259.

Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; and Yakhnenko, O. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, 2787–2795.

Bruna, J.; Zaremba, W.; Szlam, A.; and Lecun, Y. 2014. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations (ICLR2014), CBLS, April 2014*.

Cao, Y.; Liu, Z.; Li, C.; Li, J.; and Chua, T.-S. 2019a. Multi-channel graph neural network for entity alignment. *arXiv preprint arXiv:1908.09898* .

Cao, Y.; Wang, X.; He, X.; Hu, Z.; and Chua, T.-S. 2019b. Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *The world wide web conference*, 151–161.

Chakrabarti, D.; Kumar, R.; and Tomkins, A. 2006. Evolutionary clustering. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 554–560.

Chen, M.; Tian, Y.; Chang, K.-W.; Skiena, S.; and Zaniolo, C. 2018. Co-training embeddings of knowledge graphs and entity descriptions for cross-lingual entity alignment. *arXiv preprint arXiv:1806.06478* .

Chen, M.; Tian, Y.; Yang, M.; and Zaniolo, C. 2016. Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. *arXiv preprint arXiv:1611.03954* .

Dasgupta, S. S.; Ray, S. N.; and Talukdar, P. 2018. Hyte: Hyperplane-based temporally aware knowledge graph embedding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2001–2011.

Grover, A.; and Leskovec, J. 2016. node2vec: Scalable Feature Learning for Networks. *CoRR* abs/1607.00653. URL http://arxiv.org/abs/1607.00653.

Guo, L.; Sun, Z.; and Hu, W. 2019. Learning to exploit long-term relational dependencies in knowledge graphs. *arXiv preprint arXiv:1905.04914* .

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, 1024–1034.

Hao, Y.; Zhang, Y.; He, S.; Liu, K.; and Zhao, J. 2016. A joint embedding method for entity alignment of knowledge bases. In *China Conference on Knowledge Graph and Semantic Computing*, 3–14. Springer.

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* .

Lao, N.; and Cohen, W. W. 2010. Relational retrieval using a combination of path-constrained random walks. *Machine learning* 81(1): 53–67.

Lehmann, J.; Isele, R.; Jakob, M.; Jentzsch, A.; Kontokostas, D.; Mendes, P. N.; Hellmann, S.; Morsey, M.; Van Kleef, P.; Auer, S.; et al. 2015. DBpedia–a large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web* 6(2): 167–195.

Li, C.; Cao, Y.; Hou, L.; Shi, J.; Li, J.; and Chua, T.-S. 2019. Semi-supervised Entity Alignment via Joint Knowledge Embedding Model and Cross-graph Model. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2723–2732.

Mahdisoltani, F.; Biega, J.; and Suchanek, F. 2014. Yago3: A knowledge base from multilingual wikipedias. In *7th biennial conference on innovative data systems research*. CIDR Conference.

Mao, X.; Wang, W.; Xu, H.; Lan, M.; and Wu, Y. 2020. MRAEA: An Efficient and Robust Entity Alignment Approach for Cross-lingual Knowledge Graph. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, 420–428.

Nguyen, G. H.; Lee, J. B.; Rossi, R. A.; Ahmed, N. K.; Koh, E.; and Kim, S. 2018. Continuous-time dynamic network embeddings. In *Companion Proceedings of the The Web Conference 2018*, 969–976.

Nguyen, T. H.; Moreira, V. P.; Nguyen, H.; Nguyen, H.; and Freire, J. 2011. Multilingual Schema Matching for Wikipedia Infoboxes. *CoRR* abs/1110.6651. URL http://arxiv.org/abs/1110.6651.

Nickel, M.; Tresp, V.; and Kriegel, H.-P. 2011. A three-way model for collective learning on multi-relational data. In *Icml*.

Nickel, M.; Tresp, V.; and Kriegel, H.-P. 2012. Factorizing yago: scalable machine learning for linked data. In *Proceedings of the 21st international conference on World Wide Web*, 271–280.

Ou, M.; Cui, P.; Pei, J.; Zhang, Z.; and Zhu, W. 2016. Asymmetric Transitivity Preserving Graph Embedding. In *KDD*.

Ou, M.; Cui, P.; Wang, F.; Wang, J.; and Zhu, W. 2015. Non-transitive Hashing with Latent Similarity Components. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, 895–904. New York, NY, USA: ACM. ISBN 978-1-4503-3664-2. doi:10.1145/2783258.2783283. URL http://doi.acm.org/10.1145/2783258.2783283.

Pei, S.; Yu, L.; Hoehndorf, R.; and Zhang, X. 2019. Semi-supervised entity alignment via knowledge graph embedding with awareness of degree difference. In *The World Wide Web Conference*, 3130–3136.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. DeepWalk: Online Learning of Social Representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, 701–710. New York, NY, USA: ACM. ISBN 978-1-4503-2956-9. doi:10.1145/2623330.2623732. URL http://doi.acm.org/10.1145/2623330.2623732.

Rebele, T.; Suchanek, F.; Hoffart, J.; Biega, J.; Kuzey, E.; and Weikum, G. 2016. YAGO: A multilingual knowledge base from wikipedia, wordnet, and geonames. In *International Semantic Web Conference*, 177–185. Springer.

Rinser, D.; Lange, D.; and Naumann, F. 2013. Cross-lingual entity matching and infobox alignment in Wikipedia. *Information Systems* 38: 887–907. doi:10.1016/j.is.2012.10.003.

Sajjad, H. P.; Docherty, A.; and Tyshetskiy, Y. 2019. Efficient representation learning using random walks for dynamic graphs. *arXiv preprint arXiv:1901.01346* .

Savenkov, D.; and Agichtein, E. 2016. CRQA: Crowd-powered real-time automatic question answering system. In *Fourth AAAI Conference on Human Computation and Crowdsourcing*.

Sharan, U.; and Neville, J. 2008. Temporal-relational classifiers for prediction in evolving domains. In *2008 Eighth IEEE International Conference on Data Mining*, 540–549. IEEE.

Shi, X.; and Xiao, Y. 2019. Modeling Multi-mapping Relations for Precise Cross-lingual Entity Alignment. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 813–822.

Srivastava, R. K.; Greff, K.; and Schmidhuber, J. 2015. Highway Networks. *CoRR* abs/1505.00387. URL http://arxiv.org/abs/1505.00387.

Suchanek, F. M.; Abiteboul, S.; and Senellart, P. 2011. Paris: Probabilistic alignment of relations, instances, and schema. *arXiv preprint arXiv:1111.7164* .

Sun, Y.; Han, J.; Aggarwal, C. C.; and Chawla, N. V. 2012. When will it happen? relationship prediction in heterogeneous information networks. In *Proceedings of the fifth ACM international conference on Web search and data mining*, 663–672.

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

Sun, Z.; Hu, W.; Zhang, Q.; and Qu, Y. 2018. Bootstrapping Entity Alignment with Knowledge Graph Embedding. In *IJCAI*, 4396–4402.

Sun, Z.; Wang, C.; Hu, W.; Chen, M.; Dai, J.; Zhang, W.; and Qu, Y. 2019. Knowledge graph alignment network with gated multi-hop neighborhood aggregation. *arXiv preprint arXiv:1911.08936* .

Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; and Mei, Q. 2015. LINE: Large-scale Information Network Embedding. *CoRR* abs/1503.03578. URL http://arxiv.org/abs/1503.03578.

Wang, Q.; Mao, Z.; Wang, B.; and Guo, L. 2017a. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering* 29(12): 2724–2743.

Wang, S.; Tang, J.; Aggarwal, C.; Chang, Y.; and Liu, H. 2017b. Signed network embedding in social media. In *Proceedings of the 2017 SIAM international conference on data mining*, 327–335. SIAM.

Wang, Z.; Lv, Q.; Lan, X.; and Zhang, Y. 2018. Cross-lingual knowledge graph alignment via graph convolutional networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 349–357.

Wu, Y.; Liu, X.; Feng, Y.; Wang, Z.; Yan, R.; and Zhao, D. 2019a. Relation-aware entity alignment for heterogeneous knowledge graphs. *arXiv preprint arXiv:1908.08210* .

Wu, Y.; Liu, X.; Feng, Y.; Wang, Z.; and Zhao, D. 2019b. Jointly Learning Entity and Relation Representations for Entity Alignment. *arXiv preprint arXiv:1909.09317* .

Xiong, C.; Power, R.; and Callan, J. 2017. Explicit semantic ranking for academic search via knowledge graph embedding. In *Proceedings of the 26th international conference on world wide web*, 1271–1279.

Xu, H.; Bao, J.; and Zhang, G. 2020. Dynamic Knowledge Graph-based Dialogue Generation with Improved Adversarial Meta-Learning. *arXiv preprint arXiv:2004.08833* .

Xu, K.; Wang, L.; Yu, M.; Feng, Y.; Song, Y.; Wang, Z.; and Yu, D. 2019. Cross-lingual knowledge graph alignment via graph matching neural network. *arXiv preprint arXiv:1905.11605* .

Yu, M.; Yin, W.; Hasan, K. S.; Santos, C. d.; Xiang, B.; and Zhou, B. 2017. Improved neural relation detection for knowledge base question answering. *arXiv preprint arXiv:1704.06194* .

Zhou, L.; Yang, Y.; Ren, X.; Wu, F.; and Zhuang, Y. 2018. Dynamic network embedding by modeling triadic closure process. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Zhu, H.; Xie, R.; Liu, Z.; and Sun, M. 2017. Iterative Entity Alignment via Joint Knowledge Embeddings. In *IJCAI*, 4258–4264.

Zhu, Q.; Zhou, X.; Wu, J.; Tan, J.; and Guo, L. 2019. Neighborhood-aware attentional representation for multilingual knowledge graphs. In *See Proceedings of the twenty-eighth international joint conference on artificial intelligence, IJCAI*, 10–16.