Data Mining and Knowledge Discovery (2023) 37:2494–2516
https://doi.org/10.1007/s10618-023-00963-3

i-Align: an interpretable knowledge graph alignment model

Bayu Distiawan Trisedya[1,3] · Flora D. Salim[2,3] · Jeffrey Chan[3] · Damiano Spina[3] · Falk Scholer[3] · Mark Sanderson[3]

Received: 28 November 2022 / Accepted: 12 July 2023 / Published online: 9 August 2023

Abstract

Knowledge graphs (KGs) are becoming essential resources for many downstream applications. However, their incompleteness may limit their potential. Thus, continuous curation is needed to mitigate this problem. One of the strategies to address this problem is KG alignment, i.e., forming a more complete KG by merging two or more KGs. This paper proposes i-Align, an interpretable KG alignment model. Unlike the existing KG alignment models, i-Align provides an explanation for each alignment prediction while maintaining high alignment performance. Experts can use the explanation to check the correctness of the alignment prediction. Thus, the high quality of a KG can be maintained during the curation process (e.g., the merging process of two KGs). To this end, a novel Transformer-based Graph Encoder (Trans-GE) is proposed as a key component of i-Align for aggregating information from entities' neighbors (structures). Trans-GE uses *Edge-gated Attention* that combines the adjacency matrix and the self-attention matrix to learn a gating mechanism to control the information aggregation from the neighboring entities. It also uses *historical embeddings*, allowing Trans-GE to be trained over mini-batches, or smaller sub-graphs, to address the scalability issue when encoding a large KG. Another component of i-Align is a Transformer encoder for aggregating entities' attributes. This way, i-Align can generate explanations in the form of a set of the most influential attributes/neighbors based on attention weights. Extensive experiments are conducted to show the power of i-Align. The experiments include several aspects, such as the model's effectiveness for aligning KGs, the quality of the generated explanations, and its practicality for aligning large KGs. The results show the effectiveness of i-Align in these aspects.

Keywords knowledge graph · Graph pattern matching · Entity alignment · Explainability · Interpretability

Responsible editor: Charalampos Tsourakakis.

Extended author information available on the last page of the article

1 Introduction

Knowledge graphs (KGs) are getting more attention in the research community and industry. They are shown to effectively improve the performance of many downstream applications, such as question answering (Berant et al. 2013; Fader et al. 2014), recommender systems (Zhang et al. 2016; Zhao et al. 2017), and information retrieval (Ensan and Bagheri 2017; Liu and Fang 2015; Reinanda et al. 2020). Despite their usefulness, KGs are notoriously incomplete (Wang et al. 2017), and hence they require continuous curation and enrichment.

One of the most effective KG enrichment techniques is KG alignment (Paulheim 2017), which aims to merge two or more KGs to form a single and more complete KG. The first step of KG alignment is finding entities representing the same real-world entity in different KGs. Then, the relationships (i.e., the attributes and the neighbors) of the aligned entities from the different KGs are merged to form a more comprehensive KG. The former step becomes the main challenge of a KG alignment technique, while the latter is straightforward.

Traditional approaches of KG alignment mainly use string matching of entities' attributes to compute entity similarity (Volz et al. 2009; Pershina et al. 2015). These approaches require manually defined constraints, i.e., they need to know which attributes are to be compared beforehand. However, the manually defined constraints are typically sub-optimal since different entities may have different attributes, e.g., a person may have a `gender` attribute, but an organisation does not.

The state-of-the-art KG alignment approach is based on entity embeddings (Trisedya et al. 2019; Wu et al. 2020; Liu et al. 2020). To compute the entity similarity, it first computes the embeddings (a vector representation) of all entities in the KGs. Then, a vector similarity (e.g., cosine similarity) can be used as the entity similarity score. Despite its success, the practical use of embedding-based KG alignment techniques for KG enrichment is low. One of the key reasons is that these techniques do not provide any explanations of the alignment results, which is essential to help experts decide whether the alignment is correct. The expert check is important to maintain the quality of the resulting KG. Without any explanation of the prediction, an expert needs to look up all the attributes and neighbors of the aligned entities predicted by the model to verify the prediction. This can be error-prone since different KGs often have different naming schemes for attributes and relations (e.g., `educated_at` vs. `alumni`).

This paper aims to fill this gap by proposing an interpretable embedding-based KG alignment model capable of generating state-of-the-art performance with explainable alignment results. There are three main challenges to build such a model. The first is the interpretability of the two common approaches for embedding-based KG alignment techniques, Translation-based and Graph Neural Network (GNN)-based KG alignment models, is non-trivial. The translation-based alignment models compute entity embeddings using a translation-based embedding model, such as TransE (Bordes et al. 2013), that treats the triples in a KG independently, making it difficult to compute the importance of attributes

and neighbors. Meanwhile, GNN-based models (Wu et al. 2020; Liu et al. 2020) overlook entities' attributes by typically only using the entity label to initialize node embeddings in the GNN while ignoring the other attributes (e.g., birth date, address, etc.). Moreover, GNN typically employs a message-passing paradigm, where the aggregation function is constructed to be invariant to neighborhood permutations (Dwivedi and Bresson 2021). These limitations mean that the predicted alignments are difficult to explain, i.e., it is difficult to compute the importance of the attributes and neighbors of the aligned entities.

The second challenge is that applying a post-hoc (model-agnostic) explainer is sub-optimal. One of the state-of-the-art post-hoc explainers for GNN models is GNNExplainer (Ying et al. 2019). However, it can only extract the most influential neighbors, but not the most influential attributes, due to the first limitation of GNN-based models mentioned above. Moreover, model-agnostic explainers cannot have perfect fidelity with respect to the model (Rudin 2019). The third challenge is scalability. The state-of-the-art alignment models are built on top of GNN models that need to maintain the whole KB graph, which requires large amounts of memory for the message-passing procedure. This is problematic when handling large KGs.

This paper proposes an interpretable KG alignment model named *i-Align* to handle the above challenges. The main goal of i-Align is to accurately predict entity alignment between KGs and seamlessly provide an explanation for the prediction. The provided explanation is in the form of the similarity between the top-n features (i.e., attributes and neighbors) of the aligned entities used to compute the entity embeddings. Intuitively, a KG alignment model should capture the aligned features of aligned entities, which are reflected by the computed embeddings. In other words, entity embedding is computed by highlighting the features that are aligned with the features of the counterpart entity. Hence, the top-n highlighted attributes and neighbors of entities can help to indicate the correctness of the predictions.

The proposed model is built on top of a Transformer model (Vaswani et al. 2017) to exploit its self-attention mechanism to rank the importance of the attributes and neighbors (Wiegreffe and Pinter 2019) as the model's explanation. It has two Transformer encoders, one is used as an attribute aggregator, and the other is used as a neighbor aggregator. The attribute aggregator computes a hidden state of an entity based on its attribute using the standard Transformer (Vaswani et al. 2017). At the same time, the neighbor aggregator computes a hidden state based on its structure/neighbors. Both hidden states are combined to form entity embeddings.

This, however, poses an additional challenge. Computing the hidden states (latent information) based on structure/neighbors using a Transformer-based model is challenging, especially in large KGs, which leads to the following sub-problems. First, the self-attention mechanism of a Transformer is computationally expensive and may not be feasible to be applied to a large graph. Thus, the model needs to decompose the large graph into sub-graphs (mini-batches). Second, it requires a message-passing-like mechanism to aggregate the structural information of both the sub-graphs and the whole graph accordingly. Existing work, such as GraphTransformer (Dwivedi and Bresson 2021), has attempted to simulate the message-passing mechanism of a GNN in a Transformer-based model, but it cannot handle large graphs.

To address the above challenges, a novel Transformer-based Graph Encoder, *Trans-GE*, is proposed for the neighbor aggregator component of i-Align. Trans-GE uses *Edge-gated Attention* that combines the adjacency matrix and the self-attention matrix to learn a gating mechanism to control the information aggregation from neighboring entities. It also uses Historical Embeddings (Chen et al. 2018; Fey et al. 2021), which allows Trans-GE to approximate the full computational graph in a mini-batch to address the scalability issue when encoding a large KG. The attention mechanism of the attributes and neighbor aggregators is used to compute the attention weight to highlight the important attributes and neighbors, respectively. The top-n highlighted attributes and neighbors of the aligned entities are then listed as an explanation of whether the alignment is correct.

In summary, the contributions of the paper are as follows:

1. An interpretable KG alignment model is proposed, where an explanation of the alignment prediction can be automatically derived. The alignment prediction can help enrich a KG, and the explanation can help experts check the correctness of the prediction to maintain the high quality results of the enrichment process.
2. Along with the proposed model, a novel Transformer-based graph encoder is proposed. It uses Edge-gated Attention to learn a weight that controls information aggregation from the surrounding neighbors of an entity. It also uses Historical Embeddings to train the model over small mini-batches.
3. Extensive experiments and analyses are conducted to show the model's effectiveness in predicting the alignments and providing explanations.

2 Related work

In this section, first, two approaches of embedding-based KG alignment models are discussed, including translation-based and GNN-based models. Then, explanation techniques for GNN models are also discussed.

2.1 Embedding-based KG alignment

Embedding-based KG alignment models have become popular since the success of knowledge representation learning (Wang et al. 2017). Two common approaches of embedding-based KG alignment are translation-based and GNN-based models. Generally, the embedding-based KG alignment models consist of two modules. The first is the embedding module that computes the embeddings of entities and predicates of two KGs. The second is the alignment module that learns an alignment matrix to transform entity embeddings from one KG vector space to the other KG vector space so that the entity embeddings from the two KGs fall in the same vector space. The alignment module is trained using a seed set of aligned entities (*seed alignment* for short). The fundamental difference between the two is how the entity embedding is computed. Translation-based models use a translation-based KG embedding model, such as TransE (Bordes et al. 2013). In contrast, GNN-based models use a Graph

Neural Network encoder, such as Graph Convolutional Networks (GCN) (Kipf and Welling 2017).

Translation-based models MtransE (Chen et al. 2017) is the first effort on embedding-based KG alignment. It uses TransE in its embedding module and shallow networks to learn an alignment matrix. IPTransE (Zhu et al. 2017) uses an extension of TransE called PTransE, which advances TransE by considering a path between two entities in a KG triple. Another representative KG alignment is BootEA (Sun et al. 2018), which iteratively adds the seed alignment from the highly-confident predicted alignments in the previous training iteration. The alignment module of BootEA is a one-to-one classifier instead of a transformation matrix used by the predecessors. TransEdge (Sun et al. 2019) aims to improve the embedding module by considering the contextual information when computing the predicate embeddings. When processing a KG triple, it uses a function to combine the predicate embeddings with the head and tail embeddings.

The above models only consider the KG structures (entity's neighbors) and ignore the attributes. The subsequent development of embedding-based KG alignment models shows that exploiting the attributes improves the model's performance significantly. JAPE (Sun et al. 2017) is the first to study the integration of attribute information for KG alignment. It treats the attribute as an *attribute triple*, which contains an entity, a relationship/predicate, and an attribute, similar to the typical KG triple. Hence, the TransE embedding models can be directly applied in its embedding module. Further development advances JAPE by using a more robust attribute encoder. For example, AttrE (Trisedya et al. 2019) uses LSTM and N-gram embeddings to encode the attribute values, while MultiKE (Zhang et al. 2019) uses Convolution Neural Networks (LeCun et al. 2015). The other work uses additional textual information to improve the embedding module. For example, KDCoE (Chen et al. 2018) uses entity description.

GNN-based models GCN-Align (Wang et al. 2018) is the first method that uses Graph Convolutional Networks (GCN) as an embedding module. The reason is that GCN can better capture the structural information in a graph (e.g., a KG). HGCN (Wu et al. 2019) improves GCN-Align by explicitly adding predicate embeddings as additional features. MuGNN (Cao et al. 2019) further improves GCN-Align by inferring the missing relationships/predicates using the Horn rules. GMNN (Xu et al. 2019) treats KG alignment as graph matching and uses node-level and graph-level matching modules. The node-level matching is similar to the embedding module of the predecessors, while the graph-level matching uses another GCN to encode *topic graphs* contained in a KG. NMN (Wu et al. 2020) uses neighborhood differences as additional features. They propose a sampling strategy to select the most representative/informative neighbors to get the neighborhood difference.

The above-mentioned models use GCN in their embedding module. Some models use the variants of GCN, such as KECG (Li et al. 2019) uses Graph Attention Networks (GAT) (Velickovic et al. 2018). AVR-GCN (Ye et al. 2019) proposes vectorized relational GCN. SSP (Nie et al. 2021) combines TransE and GCN as its embedding module. The other work along this line uses various information as additional features. For example, AliNET (Sun et al. 2020) considers the neighbor distance, MRAEA (Mao et al. 2020) includes meta-relation, such as relation direction

and inverse relation, and AttrGNN (Liu et al. 2020) splits the attributes into different views, including attribute label, attribute literal values, and attribute digital values.

In general, the explanation of the predicted alignment from GNN-based models is difficult to obtain. The reason is that the GNN used in these models employs a message-passing paradigm, where the aggregation function is constructed to be invariant to neighborhood permutations (Dwivedi and Bresson 2021). Thus, it is difficult to quantify the importance of the neighbors for computing the entity embeddings. Moreover, GNN-based models overlook entities' attributes. Typically, they only use the entity label to initialize node embeddings in the GNN and ignore the other attributes. This makes it difficult to measure the importance of the attributes. One alternative to getting an explanation of GNN-based models is by applying a post-hoc explainer, which is discussed in the following section.

2.2 Explanation techniques for GNN

A post-hoc explainer treats a machine learning model (e.g., a GNN model) as a black box. It approximates the behavior of a model by extracting relevant information to reveal the attribution of the input features. There are two common approaches for post-hoc explainers: *model agnostic* and *model specific*. The main difference is that the model agnostic explainers do not consider the model's internal components, e.g., the learned weights. Typically, they make a black box model more transparent by creating an approximation around the prediction using a linear (but local) classifier (Ribeiro et al. 2016), feature attribution (Lundberg and Lee 2017), saliency mappings (Bach et al. 2015; Lapuschkin et al. 2019), or rule-based explanations (Bastani et al. 2017). Model agnostic approaches provide flexibility in model architecture and explanation type (e.g., linear formula or feature importance). However, they may only provide local explanations and require high computation when permuting the input features. This is problematic when dealing with large KGs that can have a huge number of entities and attributes. On the other hand, model specific explainers consider the model's internal components and mostly focus on deep neural network models. The most commonly used components are gradients (Liu and Gifford 2017; Selvaraju et al. 2017), attentions (Kumar et al. 2017), and neuron contributions (Chen et al. 2018; Shrikumar et al. 2017; Sundararajan et al. 2017). The limitation of this approach is that it binds to one type of model.

The methods mentioned above are designed for non-GNN models. Adapting these methods for explaining a GNN model is non-trivial, as recent studies show that these methods are prone to gradient saturation problems, especially when computing the gradient of a large adjacency matrix (Ying et al. 2019).

There is limited work on the post-hoc (model agnostic) explanation method of GNN models. GNNExplainer (Ying et al. 2019) extracts the most dominant sub-graph of the input graph that affects the predictions. Specifically, the sub-graph is extracted by taking N nodes that give the highest mutual information with the prediction. PGExplainer (Luo et al. 2020) improves on GNNExplainer by selectively choosing the sub-graph for the candidate explanation instead of trying all permutations of the sub-graph. GNNExplainer can highlight the top-n important neighbors in explaining the existing alignment

models. However, it cannot extract the top-n important attributes since most of the existing GNN-based KG alignment models use pre-trained (or randomly initialized) embeddings as the node features.

Our proposed model fills this gap by providing a model that can accurately predict entity alignment between KGs and seamlessly provide an explanation for the prediction. The provided explanation is in the form of the similarity between the top-n features (i.e., attributes and neighbors) of the aligned entities used to compute the entity embeddings. Our model can highlight the top-n attributes and neighbors used by the model to compute the embedding of the aligned entities from two different KGs.

3 Proposed KG alignment model

This section first discusses the formulation of the KG alignment problem. The detail of the proposed solution is described in the following subsection.

3.1 Problem formulation

A KG is an extensive repository of facts, where each fact is represented as a triple. There are two types of a triple: an attribute triple, which represents the properties of entities (e.g., `birth_date`), and a relationship triple, which represents the relation between entities (e.g., `spouse`). An attribute triple is denoted as $\langle h, r, a \rangle$, while a relationship triple is denoted as $\langle h, r, t \rangle$. Here, h is the head entity, r is the predicate (i.e., the relation or the attribute key), t is the tail entity, and a is the attribute value. The relationship triples can be represented as a neighborhood graph $\mathcal{G}$ with vertices/nodes $\mathcal{V}$ representing entities and edges $\mathcal{E}$ representing relationships.

Let $\mathcal{G}_A$ and $\mathcal{G}_B$ denote the two KGs to be aligned. The first objective is to find every pair $\langle v_a, v_b \rangle$ where $v_a \in \mathcal{G}_A$, $v_b \in \mathcal{G}_B$, and v_a and v_b represent the same real-world entity. The second objective is to provide an explanation for each extracted pair (predicted alignment). The explanation is in the form of a set of attributes and neighbors of each entity in the pair that can help experts decide whether the alignment is correct. The attributes and neighbors are selected based on their contribution in computing the entity embeddings. Here, only the most influential (e.g., top-n) attributes and neighbors are selected for the explanation. Table 3 shows an example of the explanation.

3.2 i-Align

The proposed i-Align is an embedding-based KG alignment model. It uses the vector similarity of the entity embeddings to compute the similarity between entities from two different KGs. The entity embedding is computed based on attribute triples and relationship triples as follows.

$$\mathbf{h}_v = [\mathbf{h}_v^{\text{att}}; \mathbf{h}_v^{\text{nei}}] \tag{1}$$

where $\mathbf{h}_v$ denotes the entity embedding, $\mathbf{h}_v^{\text{att}}$ denotes the attribute embeddings (a vector computed by the attribute aggregator, Sect. 3.2.1), $\mathbf{h}_v^{\text{nei}}$ denotes the neighborhood embeddings (a vector computed by the neighbor aggregator, Sect. 3.2.2), and [;] denotes a concatenation operator.

The model learns the alignment via learning close entity embeddings for the aligned entities. Specifically, it learns to separate the positive samples (the aligned entities in the seed alignment $\mathcal{S}$) from the negative sample (randomly generated) by a large margin. It uses a margin ranking loss $\mathcal{L}_{align}$ defined as follows.

$$\mathcal{L}_{\text{align}} = \sum_{s \in \mathcal{S}} \sum_{s' \in \mathcal{S}'} \max\left(0, \left[\gamma + f(s) - f(s')\right]\right) \tag{2}$$

$$f(s) = \left\| \mathbf{h}_{v_a} - \mathbf{h}_{v_b} \right\| \tag{3}$$

$$f(s') = \left\| \mathbf{h}_{v_a} - \mathbf{h}_{v'_b} \right\| \tag{4}$$

Here, $\mathcal{S}$ is a seed alignment, containing a list of aligned entity pairs from $\mathcal{G}_A$ and $\mathcal{G}_B$, i.e., $s = \langle v_a, v_b \rangle$, that acts as positive samples. $\mathcal{S}'$ is the negative samples generated by randomly changing one of the entities in each aligned entity pair of the seed alignment, e.g., v_b is changed into v'_b, a random entity in $\mathcal{G}_B$. For one positive sample, i-Align takes five negative samples.[1] The symbol γ denotes the margin hyper-parameter.

i-Align is built on top of the Transformer model to exploit its self-attention mechanism to highlight the most important attributes/neighbors as the explanation of the alignment prediction. The self-attention mechanism is computationally expensive and may not be feasible to be applied to a large graph. Thus, i-Align decomposes the large graph into sub-graphs (mini-batches). It uses edge-gated attention and historical embeddings (Sect. 3.2.2), which allows i-Align to be trained over mini-batches.

The mini-batch construction is straightforward, as illustrated in Fig. 2. First, following Cluster-GCN (Chiang et al. 2019), a graph clustering algorithm METIS (Karypis and Kumar 1998) is used to sample a small number of entities that have high inter-connectivity among them (denoted by blue circles in the figure) as the initial mini-batch sample. The clustering algorithm may generate different sizes of sub-graph in the mini-batch procedure. However, this does not affect the end results of the performance. It may affect the number of epochs to reach the convergence of the model performance. If the clustering algorithm creates large graphs, it may require more epochs to reach convergence. Next, all attributes of these entities, i.e., the attribute triples, are extracted (denoted by green rectangles). These attribute triples are then sent to the attribute aggregator to be processed. Finally, all the selected entities' first-hop neighbors (denoted by orange circles), which may not be selected in the sampling process, are extracted and merged into the mini-batch. This sub-graph is

[1] Since a KG consists of many entities, a larger negative sample may give a better alignment performance but have a trade-off in the running-time performance. Existing KG alignment models use 5-10 negative samples.

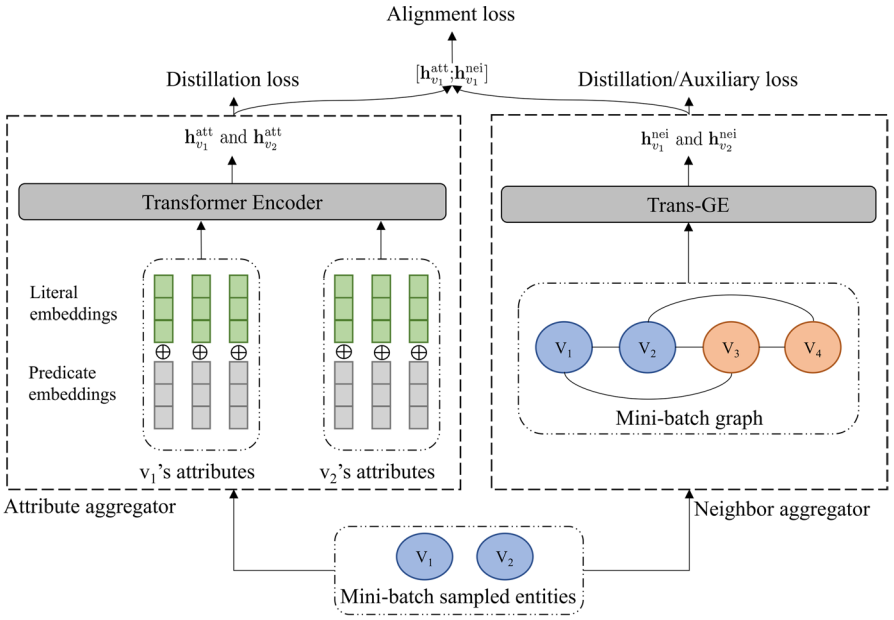

Fig. 1 The proposed model consists of two main modules. The first is an **attribute aggregator** that aims to compute node representation based on the node's attributes. The second is a **neighbor aggregator** that aims to compute node representation based on the node's neighborhood graph structure

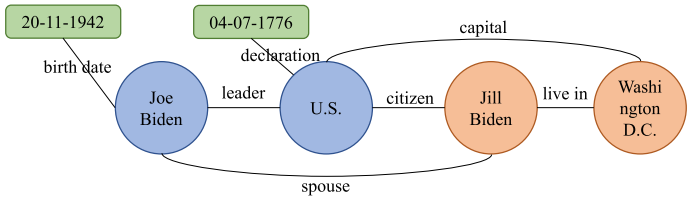

Fig. 2 A mini-batch sample. The blue circles indicate the initial mini-batch sample. The green rectangles indicate the attributes of the sampled entities. The orange circles indicate the first-hop neighbors of the sampled entities

then sent to the neighbor aggregator to be processed. The overall process of i-Align is illustrated in Fig. 1.

3.2.1 Attribute aggregator

The goal of the attribute aggregator is to compute the attribute embeddings h^{att}_v given the attribute triples in a mini-batch. For each entity in the mini-batch, the aggregator takes the corresponding attribute keys and values in the attribute triples. Note that the attribute value may be a literal (a sequence of character, i.e., $l = [c_1, c_2, ..., c_l]$), such as a person's name or birth date. Thus, before combining the attribute key and value to compute the final attribute embeddings, the aggregator uses Gated

Recurrent Unit (GRU)[2] Cho et al. (2014) to transform an attribute value into a literal embedding (Eq. 5). Next, it combines the literal embeddings and the predicate (attribute key) embeddings (Eq. 6) and uses self-attention (i.e., Transformer) to obtain the interaction between the attributes of an entity (Eq. 7–8). Formally, the attribute embeddings computation is defined as follows.

$$\mathbf{l}_v = \text{GRU}\left([\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_l]\right) \tag{5}$$

$$\mathbf{x}_v^{\text{att}} = \tanh\left(\mathbf{W}_a \mathbf{a}_v \mathbf{W}_l \mathbf{l}_v\right) \tag{6}$$

$$\alpha_{\text{att}} = \text{softmax}\left(\frac{\mathbf{Q}_{\text{att}} \mathbf{K}_{\text{att}}^{\top}}{\sqrt{d}}\right), \quad \text{where} \tag{7}$$

$$\begin{aligned} \mathbf{Q}_a &= \mathbf{x}_v^{\text{att}\top} \mathbf{W}_{\texttt{Qa}} \quad \text{and} \quad \mathbf{K}_a = \mathbf{x}_v^{\text{att}\top} \mathbf{W}_{\texttt{Ka}} \\ \mathbf{h}_v^{\text{att}} &= \alpha_{\text{att}} \mathbf{x}_v^{\text{att}\top} \mathbf{W}_{\texttt{Va}} \end{aligned} \tag{8}$$

Here, $\mathbf{a}_v$ is the predicate (attribute key) embeddings, d denotes the dimensionality of the embedding $\mathbf{x}_v^{\text{att}}$ and $\mathbf{W}$ denotes learned parameters. The symbol α_{att} denotes the attention weight of the attributes, which is used to generate the explanation. The attention aggregator uses three layers of self-attention (Eq. 7–8) with residual connections. Thus, the final attention weight used for explaining the attribute importance is the average of the attention weight from all layers.

3.2.2 Neighbor aggregator

The neighbor aggregator uses a Transformer-based Graph Encoder (Trans-GE) to encode a mini-batch sub-graph. Two main components of Trans-GE are edge-gated attention and historical embeddings. The edge-gated attention captures the structural information of a sub-graph. At the same time, the historical embeddings allow Trans-GE to approximate the full computational graph of a KG in a mini-batch.

Edge-gated Attention takes the adjacency matrix and the predicate (relation) embedding of the mini-batch sub-graph to learn the attribution of entity neighbors. It multiplies the adjacency matrix and the relation embeddings and applies a sigmoid function to learn each relation's weight (Eq. 9). This weight is used as a mask/gate γ to control the neighbors' influence (Eq. 10) on the self-attention computation (Eq. 11–12), which also captures the similarity between relations. The

[2] A different unit, such as LSTM, can be used here. A pre-trained encoder, such as BERT, can also be used to improve the model's capability to capture semantic similarity between attributes. GRU is chosen because it has a faster running time, with no significant drop in accuracy.

weight is also used to update the relation embeddings for computing the gate in the next layer (Eq. 13). Specifically, the edge-gated attention is defined as follows.

$$\gamma = \text{sigmoid}\left(\mathbf{A}\mathbf{W}_r\mathbf{r}_v\right) \tag{9}$$

$$\mathbf{x}_v^{\text{nei}} = \gamma\mathbf{x}_{\text{HE}} \tag{10}$$

$$\alpha_{\text{nei}} = \text{softmax}\left(\frac{\mathbf{Q}_{\text{nei}}\mathbf{K}_{\text{nei}}^{\top}}{\sqrt{d}}\right), \quad \text{where} \tag{11}$$

$$\begin{aligned}\mathbf{Q}_n &= \mathbf{x}_v^{\text{nei}\top}\mathbf{W}_{\texttt{Qn}} \quad \text{and} \quad \mathbf{K}_n = \mathbf{x}_v^{\text{nei}\top}\mathbf{W}_{\texttt{Kn}} \\ \mathbf{h}_v^{\text{nei}} &= \alpha_{\text{nei}}\mathbf{x}_v^{\text{nei}\top}\mathbf{W}_{\texttt{Vn}}\end{aligned} \tag{12}$$

$$\mathbf{r}'_v = \mathbf{r}_v + \gamma\mathbf{W}_{r'}\mathbf{r}_v \tag{13}$$

Here, $\mathbf{A}$ is the adjacency matrix, $\mathbf{r}_v$ and $\mathbf{r}'_v$ denote the current and updated relation embeddings, respectively, $\mathbf{x}_{\text{HE}}$ are the initial node embeddings from the historical embeddings, and $\mathbf{W}$ denotes a learned parameter. The symbol α_{nei} denotes the attention weight of the neighbors used to generate the explanation. Similar to the attribute aggregator, the neighbor aggregator uses three layers of edge-gated attention. The final attention weight used for explaining the neighbor importance is the average attention weight from all layers.

Historical embeddings are used by Trans-GE to compute the node embeddings $\mathbf{x}_{\text{HE}}$ (Eq. 10), which is an approximation of node embeddings that capture the whole computational graph in a KG, i.e., it approximates the node embeddings computation of the message-passing mechanism in a GNN. It borrows the idea of GNNAutoScale (Fey et al. 2021) that defines the historical embeddings as node embeddings acquired in the previous training iteration, capturing the computation graph. However, the Trans-GE historical embeddings differ from those in GNNAutoScale's as they use an approximation layer instead of the push/pull mechanism used by GNNAutoScale.

The push/pull mechanism of GNNAutoScale updates the historical embeddings whenever it computes new node embeddings in each layer so that in the next layer, it can pull the up-to-date node embeddings from historical embeddings. In contrast, the approximation layer of Trans-GE uses a linear transformation learned to approximate the current node embeddings based on the embeddings in the previous state. The advantage of using this approximation layer is that the back-propagation process of the training is more stable since it does not have disconnections in the computational graph due to the push/pull mechanism. The approximation layer is trained via distillation loss defined as follows.

$$\mathbf{x}_{\text{HE}} = \mathbf{W}_{\text{dist}}\mathbf{x}_0 \tag{14}$$

$$\mathcal{L}_{\text{HE}_1} = \sum_{v \in \mathcal{G}_A \cup \mathcal{G}_B} \sum_{k=1}^{K} \left[1 - \cos(\mathbf{x}_{\text{HE},v}^{k}, \mathbf{h}_{v}^{\text{nei},k})\right] \tag{15}$$

$$\mathcal{L}_{\text{HE}_2} = \sum_{v \in \mathcal{G}_A \cup \mathcal{G}_B} \left[1 - \cos(\mathbf{x}_0, \mathbf{h}_{v}^{\text{att}})\right] \tag{16}$$

where $\mathbf{x}_{\text{HE}}$ is the embeddings approximation, $\mathbf{x}_0$ denotes the stored historical embeddings that updated every iteration, $\mathcal{L}_{\text{HE}}$ is the distillation loss, and K is the number of layers in Trans-GE. Following GNNAutoScale, $\mathbf{L}_2$ regularization is applied to each output layer $\mathbf{h}_{v}^{\text{nei},k}$ to ensure the closeness of historical embeddings approximated in each layer. Lastly, another distillation loss (Eq. 16) is added to initialize the stored historical embeddings. The key idea is that the historical (node) embeddings can be approximated from attribute embeddings rather than random initialization.

4 Experiments

Three experiments are conducted to show the power of i-Align.[3] The first two experiments aim to show the effectiveness of i-Align in entity alignment and explanation generation. The last experiment aims to show the scalability of i-Align to handle large KGs.

4.1 KG alignment experiments

Given two KGs, the main objective of i-Align is to provide entity alignment prediction and explanation. Thus, the first experiment is conducted to measure the performance of i-Align in entity alignment compared to the state-of-the-art KG alignment models. Both translation-based and GNN-based approaches are used as baselines for this experiment. Among the translation-based approaches, MTransE (Chen et al. 2017), JAPE (Sun et al. 2017), MultiKE (Zhang et al. 2019) and AttrE (Trisedya et al. 2019) are selected. MTransE and JAPE are the pioneers of the translation-based approach, while MultiKE and AttrE are state-of-the-art. Among the GNN-based models, GCN-Align (Wang et al. 2018), EPEA (Wang et al. xxx), MRAEA (Mao et al. 2020), and NMN (Wu et al. 2020) are selected. GCN-align is the pioneer of the GNN-based approach, while MRAEA and NMN are state-of-the-art. We used the suggested hyperparameters of the baseline model from their papers.

[3] The code and datasets are available at https://bitbucket.org/bayudt/i-align.

4.1.1 Experiment settings

Dataset: The main dataset used for the experiments is DWY-NB (Zhang et al. 2022), which is used to evaluate the alignment performance and the explanation quality. This dataset contains two pairs of KGs: DBpedia - Wikidata (**DW-NB**) and DBpedia - YAGO (**DY-NB**). The DBpedia KG in DW-NB contains 84, 911 entities and 545 predicates, while the Wikidata KG contains 86, 116 entities and 703 predicates. There are 50, 000 aligned entities in DW-NB. The DBpedia KG in DY-NB contains 58, 858 entities and 211 predicates, while the YAGO KG contains 60, 228 entities and 91 predicates. There are 15, 000 aligned entities in DY-NB. For training, 30% of the aligned entities are used as seed alignment $\mathcal{S}$. The rest of the aligned entities are used for testing. We also use two additional datasets with different domains than the DW-NB dataset to further evaluate the alignment performance: **DBP-LGD** and **DBP-GEO**. These datasets contain aligned entities between DBpedia and LinkGeoData (Stadler et al. 2012) and Geonames,[4] respectively. DBP-LGD contains 10,000 aligned entities and ten aligned predicates from a total of 510 predicates, and DBP-GEO contains 10,000 aligned entities and ten aligned predicates from a total of 716 predicates.

Hyper-parameters: We use grid search to find the best hyper-parameters for the models. We choose the network dimensionality among $\{128, 256, 512\}$, the character embedding dimension among $\{16, 32, 64, 128\}$, the attribute and entity embedding dimension among $\{128, 256, 512\}$, the number of transformer layer among $\{2, 3, 4, 5\}$, the number of multi-head attention among $\{4, 8, 12\}$, and the margin γ among $\{1, 5, 10\}$.

The optimum hyper-parameters are as follows. Network hidden dimension is 256, character embedding dimension is 64, attribute and entity embeddings dimension are 256, the number of layers of the attributes (Transformer) and neighbor aggregators (Trans-GE) is 3, the number of heads in the multi-head attention is 8, margin loss $\gamma = 1$. When using a lower number than the one listed here, the performance of the model significantly drops. Meanwhile, using a higher number does not increase the performance significantly. Thus, we chose these hyper-parameters to get optimal results while having a good running time performance. The model performance stabilizes after 400 epochs of training.

Metric: The metric used to evaluate the performance is Hits@k, e.g., **Hits@1** and **Hits@10**. It is the standard evaluation metric for KG alignment Zhang et al. (2022). Hits@k indicates the percentage of entities with the correct aligned entity listed in the top-k prediction.

4.1.2 Results

Table 1 shows the comparisons of i-Align with the baseline. MTransE, JAPE, and GCN-Align achieve lower performance (i.e., below 25% in terms of Hits@1) since

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

Table 1 Comparisons of KG alignment models performance

Model	DW-NB		DY-NB		LGD-DBP		GEO-DBP	
	Hits@1	Hits@10	Hits@1	Hits@10	Hits@1	Hits@10	Hits@1	Hits@10
MTransE	7.88	25.75	0.08	0.68	33.59	35.76	33.14	34.75
JAPE	12.57	19.96	1.4	3.27	33.47	34.42	33.35	34.27
GCN-Align	24.76	48.52	24.36	53.43	48.57	52.74	46.12	51.32
MRAEA	81.54	85.97	73.71	78.52	78.98	83.13	72.11	75.32
NMN	84.03	88.21	75.87	80.54	78.88	82.35	75.87	80.18
MultiKE	84.06	90.05	84.97	90.84	83.12	90.55	79.33	85.22
AtrrE	87.98	95.8	90.44	94.23	84.17	92.05	86.91	92.32
Proposed	88.35	94.22	91.21	93.44	87.21	94.22	88.87	93.87

they overlook entities' attributes. Specifically, MTransE and GCN-Align ignore the attributes, while JAPE masks the attributes into their corresponding data types (e.g., integer, string, etc.). MRAEA and NMN exploit the attribute label (among the other entity attributes) and achieve substantial improvements. This shows that the attribute label is an important feature for the task. A more detailed discussion is provided in Sect. 4.2.

The top three models are MultiKE, AttrE, and i-Align. Specifically, i-Align achieves the highest score among them: 89.42 and 92.14 in terms of Hits@1 on DW-NB and DY-NB datasets, respectively. The three models exploit all entities' attributes, which boosts their performance. AttrE has a slightly higher Hits@10 score as it uses a more complex n-gram model as opposed to GRU used by i-Align for computing the literal embeddings. The proposed i-Align has a further advantage compared to MultiKE and AttrE. It can also generate an explanation for each alignment prediction to help experts decide its correctness. This can help maintain the high quality of the resultant KG in the KG enrichment process.

4.2 Alignment explanation experiments

This experiment aims to evaluate i-Align in generating alignment prediction explanations. Here, the explanation is in the form of a set of the most influential attributes and neighbors for computing the entity embeddings. Specifically, top-5 attributes and neighbors that have the highest attention weight computed by the attribute aggregator (Sect. 3.2.1) and neighbor aggregator (Sect. 3.2.2), respectively, are extracted as the explanation.

4.2.1 Experiment settings

Baseline: GNNExplainer (Ying et al. 2019), the state-of-the-art explanation model of GNN, is used as a strong baseline. The explanation generated by GNNExplainer is in the form of a small sub-graph with the highest mutual information. Here, GNNExplainer is coupled with GCN-Align (Wang et al. 2018). GCN-Align

Table 2 Manual evaluation of alignment explanation

Model	Correct prediction		Incorrect prediction	
	Prec	Conf	Prec	Conf
GCN-Align (Wang et al. 2018) + GNNExplainer (Ying et al. 2019)	0.39	0.47	0.80	0.75
i-Align (Neighbors only)	0.68	0.71	0.87	0.81
i-Align	0.95	0.90	0.93	0.93

is chosen because it purely uses GCN (Kipf and Welling 2017) so that it can be straightforwardly combined with GNNExplainer. This combination is adapted to compute five neighbors with the highest mutual information to compute the entity embeddings for alignment prediction. We emphasise that the combination of GCN-Align and GNNExplainer can only generate the neighbor set, while i-Align can generate both the attribute and neighbor sets.

Protocol: For this experiment, the dataset used is DW-NB. Fifty random samples for correct and incorrect prediction are collected (100 samples in total). We managed to get three annotators. The three annotators have studied artificial intelligence and machine learning for over ten years. They also closely work with knowledge graphs in the last three years. These three annotators are given two binary (yes/no) questions for each sample.

For the first question, the annotators are given an attributes/neighbors set (the explanation generated by the models, Table 3), and they need to indicate whether this set belongs to a correctly aligned entity pair. The second question asks the annotators whether they are sure with their answer to the first question. Three models compared for this experiment are GCN-Align+GNNExplainer, i-Align with only neighbors set explanation, and the proposed full i-Align. Each annotator is given the same 300 samples (randomly ordered), i.e., 100 samples for each model. The data given to the annotators is only the explanation, with no indication which model the sample belongs to. The metrics used for this evaluation are precision and confidence. Precision is the percentage of the correct answers (i.e., whether the given set belongs to a correct aligned entity pair) made by the annotators, while confidence is the number of answers where the annotators are sure of their answers. Fleiss' Kappa score is used to show the inter-rater agreement.

4.2.2 Results

Table 2 shows the results. Overall, i-Align achieves 0.95 and 0.90 in terms of precision and confidence, respectively, which are the highest scores among the tested models. The inter-rater agreement score is 0.81 based on Fleiss' Kappa, which indicated high agreement between the annotators. Providing both the influential attributes and neighbors benefits i-Align, which is shown by the substantial improvement

Table 3 Example of attributes/neighbors explanation of A **Wrong** Alignment Prediction. Both methods give a wrong alignment result of the entity *Carl Ferdinand Cori*. GCN Align + GNNExplainer provides the top-5 neighbors considered by the model to make the alignment. i-Align provides both top-5 attributes and neighbors considered by the model to make the alignment. The top-5 attributes provided by i-Align make the manual check easier

	i-Align Prediction		GCN-Align + GNNExplainer Prediction	
Entities	Carl Ferdinand Cori	Ferdinand I of Bulgaria	Carl Ferdinand Cori	Tomáš Cihlár
Attributes	(given_name, Ferdinand)	(given_name, Ferdinand)	N/A	N/A
	(date_of_death, 1984-10-20)	(date_of_death, 1948-09-10)		
	(married, 1920)	(position_end_time, 1918)		
	(last_name, Cori)	(last_name, Bulgaria)		
	(date_of_birth, 1896-12-05)	(date_of_birth, 1861-02-26)		
Neighbors	(occupation, biochemist)	(occupation, entomologist)	(occupation, biochemist)	(occupation, biochemist)
	(citizenship, Czechoslovakia)	(native_language, German)	(language, English)	(employer, Gilead Sciences)
	(place_of_death, Cambridge)	(place_of_birth, Vienna)	(gender, male)	(country_of_citizenship, Czech Republic)
	(member_of, Royal Society)	(place_of_burial, St. Augustine's Church)	(citizenship, Czechoslovakia)	(language, English)
	(place_of_birth, Prague)	(member_of, Academy of Sciences)	(employer, Harvard University)	(gender, male)

Table 4 Automatic evaluation of alignment explanation

Model	Jaccard similarity	
	Attributes	Neighbors
GCN-Align (Wang et al. 2018)	N/A	0.26
+ GNNExplainer (Ying et al. 2019)		
i-Align	0.48	0.34

compared to when only the neighbors set of i-Align or even the baseline (i.e., the combination of GCN-Align and GNNExplainer) is used. Table 3 shows the top-5 attributes/neighbors of a wrong alignment prediction (i.e., i-Align aligns the entity `Carl Ferdinand Cori` with `Ferdinand I of Bulgaria`, while GCN-Align aligns it with `Tomáš Cihlár`). Here, i-Align provides a more comprehensive explanation by providing both the attribute and neighbor alignment. In contrast, GCN-Align+GNNExplainer can not provide attribute alignment as they only use structural information for computing the entity alignment. The comprehensive explanation by i-Align makes the curation process (i.e., deciding whether the given sample is an incorrect alignment prediction made by the model) a lot easier.

4.2.3 Discussion

To further analyze the alignment explanation, a semi-automatic evaluation is performed as follows. For each tested model, 100 random correct alignment predictions are collected along with their explanations (a set of influential attributes/neighbors). Next, the explanations are pre-processed by manually changing the attributes/relations and attribute values/entity tails that refer to the same meaning, e.g., the attribute `full_name` and `name` are changed into `name`. Finally, Jaccard similarity is used to compute the set similarity. A higher score means more of the same attributes/neighbors listed in the set.

The results in Table 4 confirm the manual evaluation where i-Align provides a better explanation than the baseline. The results show that i-Align achieves a higher set similarity score, meaning that i-Align shows more aligned attributes/neighbors to explain a correct alignment prediction. This can help experts to decide the correctness of the prediction easily. Further, i-Align is more efficient in runtime performance. In the KG alignment setting, the combination of GCN-Align and GNNExplainer is expensive. It needs to compute all the permutations of neighbors to find the maximum mutual information. Moreover, for each permutation, it needs to recompute the embeddings and the alignments.

4.2.4 i-Align Behavior

Another analysis is performed to investigate the behavior of i-Align, specifically, the effect of removing attributes/neighbors. The proposed model is run five times, and

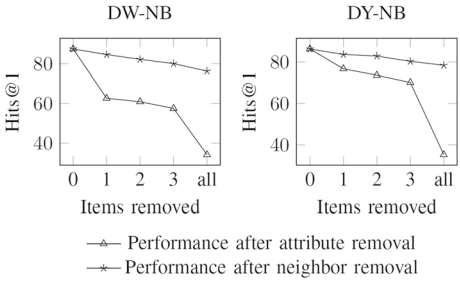

Fig. 3 The effect of attributes/neighbors removal on alignment performance

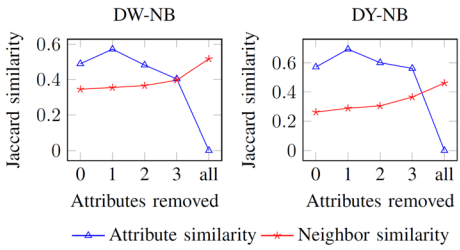

Fig. 4 The effect of attribute removal on set similarity

for each run, the most influential attributes (or neighbors) are selected based on their attention weight. For the next run, the selected attributes (or neighbors) are removed from the entities, and the entity embeddings are computed based on the remaining attributes (or neighbors) for predicting the alignments. All attributes (or neighbors) are removed for the last run, i.e., the model will run based on attributes or neighborhood only. For analysis, 100 aligned entity pairs that are correctly predicted in all runs are collected. This is done to see the change of feature importance used to make the correct prediction. The metrics used for this evaluation are Hits@1 (computed based on all test data) and Jaccard similarity score (computed based on the sampled test data).

Figure 3 shows the effect of attributes/neighbors removal on the model's performance for entity alignment. As expected, removing influential attributes/neighbors decreases the model's performance. The removal of attributes is more significantly affects the model's performance. Removing all attributes drops the performance below 40%. In comparison, the model's performance is still above 70% when removing all the neighbors. The most significant drop occurs in the second run when removing the most influential attributes. In most cases (97%), the most influential attribute is the entity name. This is in line with recent findings (Zhang et al. 2022), which show that entity name becomes a dominant feature in KG alignment.

Figures 4 and 5 show the effect of attributes/neighbor removal on the importance of attributes/neighbors by the model. In general, removing an influential attribute increases the model's attention to neighborhood similarity and vice versa. Interestingly, removing entity name, which is the dominant feature, increases both the attribute and neighbor similarity, i.e., more aligned attributes/neighbors found in the generated explanation. This means the model tries to find more evidence when there is no dominant feature.

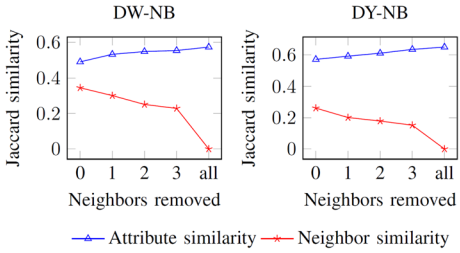

Fig. 5 The effect of neighbor removal on set similarity

Table 5 Scalability experiments results

Model	DW300K		DW600K	
	Hits@1	Running time (days)	Hits@1	Running time (days)
AttrE (Trisedya et al. 2019)	70.59	5.6	61.22	10.9
MultiKE (Zhang et al. 2019)	69.56	6.0	61.42	11.5
i-Align	72.26	5.9	63.37	11.2

4.3 Scalability

The last experiment evaluates the scalability of the proposed model. GNN-based KG alignment models have been shown to be effective in capturing both attributes and neighbors/graph structural similarity. However, the existing GNN-based models are constrained by memory usage when aligning large KGs. The proposed i-Align handles this problem by using historical embeddings.

The datasets used for this experiment are DW300K and DW600K (Zhang et al. 2022), which are three times and six times bigger than DW-NB, respectively. The baselines for this experiment are AttrE (Trisedya et al. 2019) and MultiKE (Zhang et al. 2019). They belong to the translation-based models. No GNN-based models are used as a baseline since they cannot run on the machine used for experiments because of out-of-memory error. The workstation used for this experiment has a 2.20GHz processor, 128GB main memory, and a GPU with 32GB memory.

Table 5 shows the scalability experiments result. The proposed i-Align achieves better performance than the baseline with a reasonable running time. The results show the power of i-Align in the effectiveness and practicality of aligning large KGs. Specifically, this confirms the effectiveness of the historical embeddings of i-Align. Note that the baseline is Translation-based models that cannot produce explanations for alignment prediction.

5 Conclusion and future work

This paper proposed i-Align, an interpretable KG alignment model. The main advantage of i-Align over the existing KG alignment models is that it provides an explanation for each alignment prediction made. This explanation can help experts in the curation process to merge KGs by providing an explanation of proposed alignment predictions. Thus, it helps maintain the high quality of the enriched KG. The proposed model has two components: attribute aggregator and neighbor aggregator. The attribute aggregator uses the standard Transformer, while a novel Transformer-based Graph Encoder (Trans-GE) is proposed for the neighbor aggregator. Trans-GE uses *Edge-gated Attention* that combines the adjacency matrix and the self-attention matrix to learn a score as a gate to control the information aggregation from the neighboring entities. It also uses *historical embeddings*, allowing Trans-GE trained over mini-batches/small sub-graphs to address the scalability issue when encoding a large KG. The attention mechanisms of the attribute and neighbor aggregators are used to compute the attention weight to highlight the important attributes and neighbors, respectively. Experimental results show the model's effectiveness for aligning KGs, the quality of the generated explanations, and the practicality for aligning large KGs.

The proposed i-Align uses attention weights as the primary indicator of the importance of attributes/neighbors. This is a simple yet effective technique. One limitation of i-Align is that it uses attribute literal similarity. So, it may not perform well on cross-lingual knowledge graph alignment, where the literal attributes are written in different characters. The other interesting topics for future work are: an integration to a more advanced technique, such as attention rollout (Chefer et al. 2021), which can be explored to improve the explanation quality; explainability in GNNs is an emerging research topic and worth exploring to improve i-Align.

Acknowledgements This research is supported by Australian Research Council (ARC) Centre of Excellence for Automated Decision-Making and Society (*ARC CE200100005*).

Funding Open Access funding enabled and organized by CAUL and its Member Institutions.

References

Bach S, Binder A, Montavon G, Klauschen F, Müller K-R, Samek W (2015) On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PLOS ONE 10(7):0130140

Bastani O, Kim C, Bastani H (2017) Interpretability via model extraction. In: Proceedings of FAT ML, pp 57–61
Berant J, Chou A, Frostig R, Liang P (2013) Semantic parsing on freebase from question-answer pairs. In: Proceedings of EMNLP, pp 1533–1544
Bordes A, Usunier N, Garcia-Durán A, Weston J, Yakhnenko O (2013) Translating embeddings for modeling multi-relational data. In: Proceedings of NeurIPS, pp 2787–2795
Cao Y, Liu Z, Li C, Li J, Chua T-S (2019) Multi-channel graph neural network for entity alignment. In: Proceedings of ACL, pp 1452–1461
Chefer H, Gur S, Wolf L (2021) Generic attention-model explainability for interpreting bi-modal and encoder-decoder transformers. In: Proceedings of ICCV, pp 397–406
Chen J, Song L, Wainwright M, Jordan M (2018) Learning to explain: an information-theoretic perspective on model interpretation. In: Proceedings of ICML, pp 883–892
Chen M, Tian Y, Chang K-W, Skiena S, Zaniolo C (2018) Co-training embeddings of knowledge graphs and entity descriptions for cross-lingual entity alignment. In: Proceedings of IJCAI, pp 3998–4004
Chen M, Tian Y, Yang M, Zaniolo C (2017) Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. In: Proceedings of IJCAI, pp 1511–1517
Chen J, Zhu J, Song L (2018) Stochastic training of graph convolutional networks with variance reduction. In: Proceedings of ICML, pp 942–950
Chiang W-L, Liu X, Si S, Li Y, Bengio S, Hsieh C-J (2019) Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In: Proceedings of KDD, pp 257–266
Cho K, van Merriënboer B, Bahdanau D, Bengio Y (2014) On the properties of neural machine translation: encoder–decoder approaches. In: Proceedings of the Workshop on Syntax, Semantics and Structure in Statistical Translation, pp 103–111
Dwivedi VP, Bresson X (2021) A generalization of transformer networks to graphs. In: Proceedings of AAAI
Ensan F, Bagheri E (2017) Document retrieval model through semantic linking. In: Proceedings of WSDM, pp 181–190
Fader A, Zettlemoyer L, Etzioni O (2014) Open question answering over curated and extracted knowledge bases. In: Proceedings of KDD, pp 1156–1165
Fey M, Lenssen JE, Weichert F, Leskovec J (2021) GNNAutoScale: scalable and expressive graph neural networks via historical embeddings. In: Proceedings of ICML, pp 3294–3304
Karypis G, Kumar V (1998) A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM J Sci Comput 20(1):359–392
Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. In: Proceedings of ICLR
Kumar D, Wong A, Taylor GW (2017) Explaining the unexplained: A class-enhanced attentive response (clear) approach to understanding deep neural networks. In: Proceedings of CVPR, pp 36–44
Lapuschkin S, Wäldchen S, Binder A, Montavon G, Samek W, Müller K-R (2019) Unmasking clever hans predictors and assessing what machines really learn. Nat Commun 10(1):1–8
LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436–444
Li C, Cao Y, Hou L, Shi J, Li J, Chua T-S (2019) Semi-supervised entity alignment via joint knowledge embedding model and cross-graph model. In: Proceedings of EMNLP, pp 2723–2732
Liu X, Fang H (2015) Latent entity space: a novel retrieval approach for entity-bearing queries. Inform Retriev J 18(6):473–503
Liu Z, Cao Y, Pan L, Li J, Chua T-S (2020) Exploring and evaluating attributes, values, and structure for entity alignment. In: Proceedings of EMNLP, pp 6355–6364
Liu G, Gifford D (2017) Visualizing feature maps in deep neural networks using deepresolve. a genomics case study. In: Proceedings of the ICML Workshop on Visualization for Deep Learning, pp 32–41
Lundberg SM, Lee S-I (2017) A unified approach to interpreting model predictions. In: Proceedings of NeurIPS, pp 4768–4777
Luo D, Cheng W, Xu D, Yu W, Zong B, Chen H, Zhang X (2020) Parameterized explainer for graph neural network. In: Proceedings of NeurIPS, pp 19620–19631
Mao X, Wang W, Xu H, Lan M, Wu Y (2020) Mraea: an efficient and robust entity alignment approach for cross-lingual knowledge graph. In: Proceedings of WSDM, pp 420–428
Nie H, Han X, Sun L, Wong CM, Chen Q, Wu S, Zhang W (2021) Global structure and local semantics-preserved embeddings for entity alignment. In: Proceedings of IJCAI, pp 3658–3664
Paulheim H (2017) Knowledge graph refinement: a survey of approaches and evaluation methods. Semantic web 8(3):489–508

Pershina M, Yakout M, Chakrabarti K (2015) Holistic entity matching across knowledge graphs. In: Proceedings of Big Data, pp 1585–1590
Reinanda R, Meij E, de Rijke M (2020) Knowledge graphs: an information retrieval perspective. Found Trends Inform Retriev 14(4):289–444
Ribeiro MT, Singh S, Guestrin C (2016) "why should i trust you?": explaining the predictions of any classifier. In: Proceedings of KDD, pp 1135–1144
Rudin C (2019) Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nat Mach Intell 1(5):206–215
Selvaraju RR, Cogswell M, Das A, Vedantam R, Parikh D, Batra D (2017) Grad-cam: Visual explanations from deep networks via gradient-based localization. In: Proceedings of ICCV, pp 618–626
Shrikumar A, Greenside P, Kundaje A (2017) Learning important features through propagating activation differences. In: Proceedings of ICML, pp 3145–3153
Stadler C, Lehmann J, Hoffner K, Auer S (2012) Linkedgeodata: a core for a web of spatial open data. Semantic Web 3(4):333–354
Sundararajan M, Taly A, Yan Q (2017) Axiomatic attribution for deep networks. In: Proceedings of ICML, pp 3319–3328
Sun Z, Huang J, Hu W, Chen M, Guo L, Qu Y (2019) Transedge: translating relation-contextualized embeddings for knowledge graphs. In: Proceedings of ISWC, pp 612–629
Sun Z, Hu W, Li C (2017) Cross-lingual entity alignment via joint attribute-preserving embedding. In: Proceedings of ISWC, pp 628–644
Sun Z, Hu W, Zhang Q, Qu Y (2018) Bootstrapping entity alignment with knowledge graph embedding. In: Proceedings of IJCAI, pp 4396–4402
Sun Z, Wang C, Hu W, Chen M, Dai J, Zhang W, Qu Y (2020) Knowledge graph alignment network with gated multi-hop neighborhood aggregation. In: Proceedings of AAAI, pp 222–229
Trisedya BD, Qi J, Zhang R (2019) Entity alignment between knowledge graphs using attribute embeddings. In: Proceedings of AAAI, pp 297–304
Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. In: Proceedings of NeurIPS, pp 5998–6008
Veličković P, Cucurull G, Casanova A, Romero A, Liò P, Bengio Y (2018) Graph attention networks. In: Proceedings of ICLR
Volz J, Bizer C, Gaedke M, Kobilarov G (2009) Discovering and maintaining links on the web of data. In: Proceedings of ISWC, pp 650–665
Wang Q, Mao Z, Wang B, Guo L (2017) Knowledge graph embedding: a survey of approaches and applications. IEEE Trans Knowl Data Eng 29(12):2724–2743
Wang Z, Lv Q, Lan X, Zhang Y (2018) Cross-lingual knowledge graph alignment via graph convolutional networks. In: Proceedings of EMNLP, pp 349–357
Wang Z, Yang J, Ye X Knowledge graph alignment with entity-pair embedding. In: Proceedings of EMNLP, pp 1672–1680
Wiegreffe S, Pinter Y (2019) Attention is not not explanation. In: Proceedings of EMNLP, pp 11–20
Wu Y, Liu X, Feng Y, Wang Z, Zhao D (2019) Jointly learning entity and relation representations for entity alignment. In: Proceedings of EMNLP, pp 240–249
Wu Y, Liu X, Feng Y, Wang Z, Zhao D (2020) Neighborhood matching network for entity alignment. In: Proceedings of ACL, pp 6477–6487
Xu K, Wang L, Yu M, Feng Y, Song Y, Wang Z, Yu D (2019) Cross-lingual knowledge graph alignment via graph matching neural network. In: Proceedings of ACL, pp 3156–3161
Ye R, Li X, Fang Y, Zang H, Wang M (2019) A vectorized relational graph convolutional network for multi-relational network alignment. In: Proceedings of IJCAI, pp 4135–4141
Ying R, Bourgeois D, You J, Zitnik M, Leskovec J (2019) Gnnexplainer: generating explanations for graph neural networks. In: Proceedings of NeurIPS, pp 9240–9251
Zhang Q, Sun Z, Hu W, Chen M, Guo L, Qu Y (2019) Multi-view knowledge graph embedding for entity alignment. In: Proceedings of IJCAI, pp 5429–5435
Zhang R, Trisedya BD, Li M, Jiang Y, Qi J (2022) A benchmark and comprehensive survey on knowledge graph entity alignment via representation learning. VLDB J 31(5):1143–1168
Zhang F, Yuan NJ, Lian D, Xie X, Ma W-Y (2016) Collaborative knowledge base embedding for recommender systems. In: Proceedings of KDD, pp 353–362
Zhao H, Yao Q, Li J, Song Y, Lee DL (2017) Meta-graph based recommendation fusion over heterogeneous information networks. In: Proceedings of KDD, pp. 635–644

Zhu H, Xie R, Liu Z, Sun M (2017) Iterative entity alignment via joint knowledge embeddings. In: Proceedings of IJCAI, pp 4258–4264

Authors and Affiliations

Bayu Distiawan Trisedya[1,3] · Flora D. Salim[2,3] · Jeffrey Chan[3] · Damiano Spina[3] · Falk Scholer[3] · Mark Sanderson[3]

✉ Bayu Distiawan Trisedya
bayudt@gmail.com

Flora D. Salim
flora.salim@unsw.edu.au

Jeffrey Chan
jeffrey.chan@rmit.edu.au

Damiano Spina
damiano.spina@rmit.edu.au

Falk Scholer
falk.scholer@rmit.edu.au

Mark Sanderson
mark.sanderson@rmit.edu.au

[1] Universitas Indonesia, Depok, West Java, Indonesia

[2] University of New South Wales, Sydney, NSW, Australia

[3] RMIT University, Melbourne, VIC, Australia