Exploring and Evaluating Attributes, Values, and Structures for Entity Alignment

Zhiyuan Liu[12] **Yixin Cao**[2*] **Liangming Pan**[12]
Juanzi Li[3] **Zhiyuan Liu**[3] **Tat-Seng Chua**[2]
[1]NUS Graduate School for Integrative Sciences and Engineering
[2]School of Computing, National University of Singapore, Singapore
[3]Department of CST, Tsinghua University, Beijing, China
{acharkq, caoyixin2011}@gmail.com, e0272310@u.nus.edu
{lijuanzi, liuzy}@tsinghua.edu.cn, dcscts@nus.edu.sg

Abstract

Entity alignment (EA) aims at building a unified Knowledge Graph (KG) of rich content by linking the equivalent entities from various KGs. GNN-based EA methods present promising performance by modeling the KG structure defined by relation triples. However, attribute triples can also provide crucial alignment signal but have not been well explored yet. In this paper, we propose to utilize an attributed value encoder and partition the KG into subgraphs to model the various types of attribute triples efficiently. Besides, the performances of current EA methods are overestimated because of the name-bias of existing EA datasets. To make an objective evaluation, we propose a hard experimental setting where we select equivalent entity pairs with very different names as the test set. Under both the regular and hard settings, our method achieves significant improvements (5.10% on average Hits@1 in DBP15k) over 12 baselines in cross-lingual and monolingual datasets. Ablation studies on different subgraphs and a case study about attribute types further demonstrate the effectiveness of our method. Source code and data can be found at https://github.com/thunlp/explore-and-evaluate.

1 Introduction

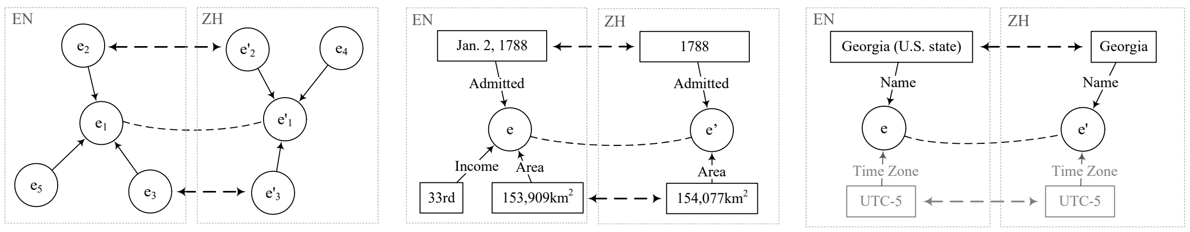
The prosperity of data mining has spawned Knowledge Graphs (KGs) in many domains that are often complementary to each other. Entity Alignment (EA) provides an effective way to integrate the complementary knowledge in these KGs into a unified KG by linking equivalent entities, thus benefiting knowledge-driven applications such as Question Answering (Yang et al., 2017, 2018), Recommendation (Cao et al., 2019b) and Information Extraction (Kumar, 2017; Cao et al., 2018). However, EA is a non-trivial task that it could be formulated as a quadratic assignment problem (Yan et al., 2016), which is NP-complete (Garey and Johnson, 1990).

A KG comprises a set of triples, with each triple consisting of a *subject*, *predicate*, and *object*. There are two types of triples: (1) *relation triples*, in which both the subject and object are entities, and the predicate is often called *relation* (see Figure 1(a)); and (2) *attribute triples*, in which the subject is an entity and the object is a *value*, which is either a number or literal string (see Figure 1(c)), and the predicate is often called *attribute*.

Most of the previous EA models (Sun et al., 2017; Wang et al., 2018; Wu et al., 2019a) rely on the structure assumption that, the adjacencies of two equivalent entities in KGs usually contain equivalent entities (Wang et al., 2018) (see Figure 1(a)). These models mainly focus on modeling KG structure defined by the relation triples. However, we argue that attribute triples can also provide important clues for judging whether two entities are the same, based on the attribute assumption that: *equivalent entities often share similar attributes and values in KGs.* For example, in Figure 1(b), the equivalent entities e and e' share the attribute *Area* with similar values of $153,909$ and $154,077$. Therefore, we aim to improve EA using attribute triples. We have identified the challenges of attribute incorporation and dataset bias.

Attribute Incorporation Challenge. Modeling attribute triples together with relation triples is a more effective strategy than modeling attribute triples alone. In this way, the alignment signal from attribute triples can be propagated to an entity's neighbors via relation triples. Recently, some pioneer EA works (Zhang et al., 2019; Trisedya et al., 2019) have incorporated both attribute and relation triples. However, they learn relation and attribute triples in separate networks. In this case, the alignment signal from an entity's discriminative attributes and values will be reserved to the

[*]Corresponding author.



(a) EA by the structure assumption. (b) EA by the attribute assumption. (c) EA with attribute importance.

Figure 1: Examples for EA using different assumptions and identifying the different importance of attributes. In Figure 1(a), we align e_1 and e'_1 for the equivalent entity pairs (e_2, e'_2) and (e_3, e'_3) in their neighbors. In Figure 1(b),1(c), we align e and e' for their similar attributes and values; e refers to the entity "Georgia (U.S. state)" from English Wiki and e' is the Chinese equivalent. In Figure 1(c), attribute *Time Zone* and its value is assigned less attention weight for being less discriminative for alignment. Chinese texts are translated. Dashed curves link the target equivalent entity pairs. Dashed bothway arrows indicate alignment signals.

entity itself and will not help align its neighbors. In addition, it is crucial to identify the different importance of attributes in discriminating whether two entities are equivalent. For example, the attribute *Time Zone* should be assigned less importance than *Name* since many cities can share the same *Time Zone* (Figure 1(c)). Previous works fail to consider the different importance of attributes.

Dataset Bias Challenge. The performance of EA is overestimated because the existing EA datasets are biased to the attribute *Name*: $60\% - 80\%$ of the released seed set of equivalent entities in DBP15k can be aligned via name matching. The reason is that the equivalent entities are collected using inter language links, which are labeled by a strategy that heavily relies on the translation of entity names[1]. In this way, the datasets contain many "easy" equivalent entities that have similar names. However, in the practical application of EA, the "easy" equivalent entities are often aligned already, and the challenge is to align the "hard" ones that have very different names. This discrepancy between datasets and practical situation causes overestimated EA performance.

To address the first challenge, we propose **Attr**ibuted **G**raph **N**eural **N**etwork (AttrGNN) to learn attribute triples and relation triples in a unified network, and learn importance of each attributes and values dynamically. Specifically, we propose an *attributed value encoder* to select and aggregate alignment signal from informative attributes and values. We further employ the mean aggregator (Hamilton et al., 2017) to propagate this signal to entity's neighbors. In addition, as different types of attributes have different similarity measurements, we partition the KG into four subgraphs by grouping attributes, *i.e.*, attribute *Name*, literal attribute, digital attribute, and structural knowledge. We apply separate channels to learn their representations. We present two methods to ensemble the outputs from all channels.

To alleviate the name-bias of EA datasets (second challenge), we propose a hard experimental setting. Specifically, we construct harder test sets from existing datasets by selecting equivalent entities that have the least similarity in their names. We further evaluate the models on these harder test sets to offer a more objective evaluation of EA models' performance. Under both the hard and regular settings, AttrGNN achieves the best result with significant performance improvement (5.10% Hits@1 on average in DBP15k) over 12 baselines on both the cross-lingual and monolingual datasets.

2 Related Work

Recent entity alignment methods can be classified into embedding-based methods and Graph Neural Network-based (GNN-based) methods.

2.1 Embedding-based Methods

Recent works utilize KG embedding methods, such as TransE (Bordes et al., 2013), to model the relation triples and further unifies two KG embedding spaces by forcing seeds to be close (Chen et al., 2017). Attribute triples has been introduced in this field. JAPE (Sun et al., 2017) computes attribute similarity to regularize the structure-based optimization. KDCoE (Chen et al., 2018) cotrains entity description and structure embeddings with a shared iteratively enlarged seed set. At-

[1]https://en.wikipedia.org/wiki/Help:Interlanguage_links

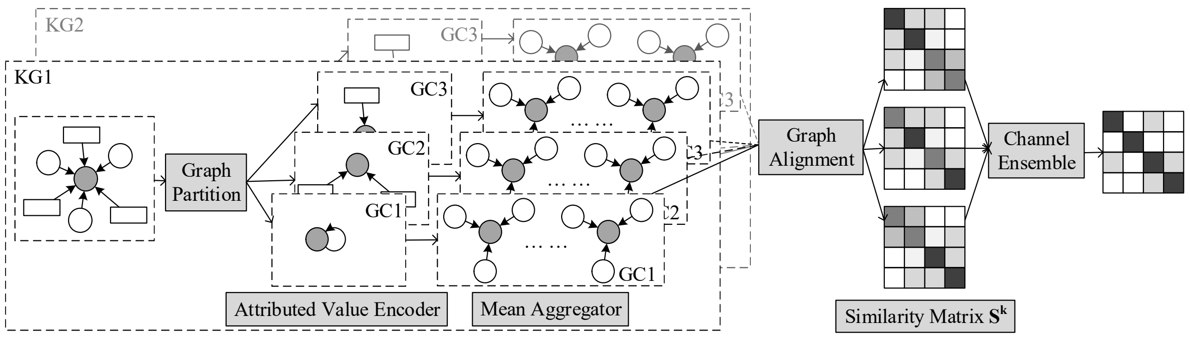


Figure 2: The framework of AttrGNN. Three GNN channels (GCs) are shown as an example. We do not use any attributes in GC1 to focus on the learning of structural knowledge (node degree distribution). $\mathbf{S}^k$ is the output similarity matrix of GCk. $\mathbf{S}^k_{e,e'}$ is the similarity between $e \in$ KG1 and $e' \in$ KG2 measured by GCk. For the KGs and its subgraphs, we use circles to denote entities and rectangles to denote values.

trE (Trisedya et al., 2019) and MultiKE (Zhang et al., 2019) encode values as extra entity embeddings. However, the diversity of attributes and uninformative values limit the performance of the above methods.

2.2 GNN-based Methods

Following Graph Convolutional Networks (Kipf and Welling, 2017), many GNN-based models are proposed because of GNN's strong ability to model graph structure. These methods present promising results on EA because GNN can propagate the alignment signal to the entity's distant neighbors. Previous GNN-based methods focus on extending GNN's ability to model relation types (Wu et al., 2019a,b; Li et al., 2019), aligning entities via matching subgraphs (Xu et al., 2019; Wu et al., 2020), and reducing the heterogeneity between KGs (Cao et al., 2019a). With the exception of Wang et al. (2018) that have incorporated attributes as the initial feature of entities, most of the current GNN-based methods fail to incorporate the attributes and values to further improve the performance of EA.

In this paper, we add values as nodes into graph and use an attributed value encoder to conduct attribute-aware value aggregation.

3 Methodology

The key idea of AttrGNN is to use graph partition and attributed value encoder to deal with various types of attribute triples. In this section, we first define KG and then introduce our graph partition strategy. Further, we design different GNN channels for different subgraphs and present two methods to ensemble all channels' outputs for final evaluation.

3.1 Model Framework

Knowledge Graph (KG) is formalized as a 6-tuple directed graph $G = (E, R, A, V, T^r, T^a)$ where E, R, A, and V refer to the set of entities, relations, attributes, and values, respectively. $T^r = \{(h, r, t) \mid h, t \in E, r \in R\}$ and $T^a = \{(e, a, v) \mid e \in E, a \in A, v \in V)\}$ is the set of relation triples and attribute triples.

Entity Alignment is to find a mapping between two KGs G and G', *i.e.*, $\psi = \{(e, e') \mid e \in E, e' \in E'\}$, where e and e' are equivalent entities. A seed set of equivalent entities ψ^s is used as training data.

Framework. The framework of our AttrGNN model is shown in Figure 2, which consists of four major components: (1) *Graph Partition*, which divides the input KG into subgraphs by grouping attributes and values. (2) *Subgraph Encoder*, which employs multiple GNN channels to learn the subgraphs separately. Each channel is a stack of L attributed value encoders and mean aggregators. The attributed value encoder aggregate attributes and values to generate the entity embeddings, and the mean aggregator propagates entity features to its neighbors following the graph structure. (3) *Graph Alignment*, which unifies the entity vector spaces of two KGs for each channel. (4) *Channel Ensemble*, which infers the entity similarity using each channel and ensemble all channels' results for final inference.

3.2 Graph Partition

Attributes and values have various types, *e.g.*, strings $\mathbb{S}$ and numbers $\mathbb{R}$. Different attributes have different similarity measurements, for example, the similarity between digital values should be numerical differences ($153,909$ *v.s.* $154,077$), while the similarity of literal values is often based on their semantic meanings. Therefore, we separately learn the similarity measurements of the KG's 4 subgraphs, defined as $G^k = (E, R, A^k, V^k, T^r, T^{ak})$, where $k \in \{1, 2, 3, 4\}$:

- G^1 includes attribute triples of *Name* only, *i.e.*, $A^1 = \{a_{name}\}$.
- G^2 includes attribute triples of literal values, *i.e.*, $A^2 = \{a \mid (e, a, v) \in T^a, v \in \mathbb{S}, a \neq a_{name}\}$.
- G^3 includes attribute triples of digital values, *i.e.*, $A^3 = \{a \mid (e, a, v) \in T^a, v \in \mathbb{R}\}$;
- G^4 has no attribute triples, *i.e.*, $A^4 = \emptyset$.

These subgraphs have mutually-exclusive attribute triples but share the same relation triples.

3.3 Subgraph Encoder

We design different GNN channels (GCs) to encode the above four subgraphs: *Name* channel for G^1, *Literal* channel for G^2, *Digital* channel for G^3, and *Structure* channel for G^4. The building blocks of these channels are two types of GNN layers: the attributed value encoder and the mean aggregator. Particularly, to select alignment signal from the informative attributes and values, we first stack one attributed value encoder and then mean aggregators in the *Literal* and *Digital* channels. We stack no attributed value encoder and only mean aggregators for the *Structure* and *Name* channels because they do not use various attribute triples. We add residual connections (He et al., 2016) between GNN layers for the *Name*, *Literal*, and *Digital* channels. Following previous EA works, all channels have two GNN layers. Next, we describe attributed value encoder and mean aggregator in details.

3.3.1 Attributed Value Encoder

Attributed value encoder can selectively gather discriminative information from the initial feature of attributes and values to the central entity. As an example, we show how to obtain e's first layer hidden state $\mathbf{h}_e^1$. The same method applies to all the entities. We obtain the sequence of attribute features $\{\mathbf{a}_1, \cdots, \mathbf{a}_n\}$ and value features $\{\mathbf{v}_1, \cdots, \mathbf{v}_n\}$ given the attribute triples $\{(e, a_1, v_1), \cdots, (e, a_n, v_n)\}$ of e as inputs. Specifically, we use BERT (Devlin et al., 2019) to obtain the features of both literal and digital values[2]. BERT is a language model that is pre-trained on a more than 3000M words corpora. It is popularly used as a feature extractor in NLP tasks. By adding values as nodes and attributes as edges, which connect values and the entity, into the graph, we then can apply attention from the entity to attributes and use the attention score to compute the weighted average of attributes and values. Following the Graph Attention Networks (Velickovic et al., 2018), we define $\mathbf{h}_e^1$ as follows:

$$\begin{aligned}
\mathbf{h}_e^1 &= \sigma\Big(\sum_{j=1}^{n} \alpha_j \mathbf{W}_1 [\mathbf{a}_j; \mathbf{v}_j]\Big), \\
\alpha_j &= \mathrm{softmax}(o_j) = \frac{\exp(o_j)}{\sum_{k=1}^{n} \exp(o_k)}, \\
o_j &= \mathrm{LeakyReLU}(\mathbf{u}^T [\mathbf{h}_e^0; \mathbf{a}_j]),
\end{aligned} \tag{1}$$

where $j \in \{1, \cdots, n\}$, $\mathbf{W}_1 \in \mathbb{R}^{D_{h_1} \times (D_a + D_v)}$ and $\mathbf{u} \in \mathbb{R}^{(D_e + D_a) \times 1}$ are learnable matrices, σ is the $\mathrm{ELU}(\cdot)$ function, and $\mathbf{h}_e^0$ is the initial entity feature.

3.3.2 Mean Aggregator

Mean aggregator layer utilizes the features of the target entity and its neighbors to generate the entity embedding. The neighbor entities of e are defined by relation triples: $\mathcal{N}(e) = \{j \mid \forall (j, r, e) \in T^r \textit{ or } \forall (e, r, j) \in T^r, \forall r \in R\}$. We aggregate the features of e's neighbor entities to gather alignment signal and learn the structural knowledge. Given the hidden state $\mathbf{h}_e^{l-1}$ from the $l-1$ layer, the mean aggregator (Hamilton et al., 2017) is defined as:

$$\mathbf{h}_e^l = \sigma(\mathbf{W}_l \, \mathrm{MEAN}(\{\mathbf{h}_e^{l-1}\} \cup \{\mathbf{h}_j^{l-1}, \forall j \in \mathcal{N}(e)\})) \tag{2}$$

where $\mathbf{W}_l \in \mathbb{R}^{D_{h_l} \times D_{h_{l-1}}}$ is a learnable matrix, $\mathrm{MEAN}(\cdot)$ returns the mean vector of the inputs, and σ is the nonlinear function chosen as $\mathrm{ReLU}(\cdot)$.

3.4 Graph Alignment

Graph Alignment unifies the two KGs' representations of each channel into a unified vector space by reducing the distance between the seed equivalent entities. We separately train the four channels and ensemble their outputs afterward for final evaluation (see Section 3.5). Following Li et al. (2019), we generate negative samples of $(e, e') \in \psi^s$ by searching the nearest entities of e (or e') in the entity embedding space. We denote the final output $\mathbf{h}_e^L$ of the channel GC^k as the entity embedding $\mathbf{e}^k$. For each channel GC^k, we optimize the following objective function:

$$\begin{aligned}
\mathcal{L}_k = \sum_{(e,e') \in \psi^s} \Big(&\sum_{e_- \in \mathrm{NS}(e)} [d(\mathbf{e}^k, \mathbf{e'}^k) - d(\mathbf{e}_-^k, \mathbf{e'}^k) + \gamma]_+ \\
&+ \sum_{e'_- \in \mathrm{NS}(e')} [d(\mathbf{e}^k, \mathbf{e'}^k) - d(\mathbf{e}, \mathbf{e'}_-^k) + \gamma]_+ \Big)
\end{aligned} \tag{3}$$

[2]As shown by Andor et al. (2019), BERT embedding can be used for simple numerical computation.

where ψ^s is the seed set of equivalent entities, $\mathrm{NS}(e)$ denotes the negative samples of e; $[\cdot]_+ = \max\{\cdot, 0\}$, $d(\cdot,\cdot) = 1 - \cos(\cdot,\cdot)$ is the cosine distance, and γ is a margin hyperparameter.

3.5 Channel Ensemble

We use the entity embedding of each channel to infer the similarity matrices $\mathbf{S}^k \in \mathbb{R}^{|E|\times|E'|}$ ($k \in \{1,2,3,4\}$), where $\mathbf{S}^k_{e,e'} = \cos(\mathbf{e}^k, \mathbf{e}'^k)$ is the cosine similarity score between $e \in E$ and $e' \in E'$. We present two methods to ensemble the four matrices into a single similarity matrix $\mathbf{S}^*$ for final evaluation.

Average Pooling. Empirically, we assume that each channel has equal importance. We let $\mathbf{S}^* = \frac{1}{4}\sum_{k=1}^{4} \tilde{\mathbf{S}}^k$, where $\tilde{\mathbf{S}}^k$ is the standardized $\mathbf{S}^k$:

$$\tilde{\mathbf{S}}^k = \frac{\mathbf{S}^k - \mathrm{mean}(\mathbf{S}^k)}{\mathrm{std}(\mathbf{S}^k)} \tag{4}$$

SVM. We utilize LS-SVM (Suykens and Vandewalle, 1999) to learn the weights for each channel: $\mathbf{S}^* = \sum_{k=1}^{4} w_k \mathbf{S}^k$, where $\mathbf{w} = [w_1, w_2, w_3, w_4]$ is trained as follow:

$$\mathcal{L}_{\text{svm}} = C\sum_{l=1}^{m}[y_l \cdot \max(0, 1 - \mathbf{w}^T\mathbf{x}_l) + (1 - y_l)\cdot \max(0, 1 + \mathbf{w}^T\mathbf{x}_l)] + \frac{1}{2}\mathbf{w}^T\mathbf{w} \tag{5}$$

where $\mathbf{x}_l = [\mathbf{S}^1_{e,e'}, \mathbf{S}^2_{e,e'}, \mathbf{S}^3_{e,e'}, \mathbf{S}^4_{e,e'}]$ is a vector of sampled similarity scores. If $(e, e') \in \phi_s$, label $y_l = 1$, otherwise $y_l = 0$.

4 Experiments

In this section, we compare AttrGNN with 12 baselines on the regular setting and our designed hard setting of EA. We also present an ablation study and a case study to evaluate attributes' and values' effects for EA.

4.1 Experimental Settings

Datasets. We test models on both cross-lingual and monolingual datasets: DBP15k (Sun et al., 2017) and DWY100k (Sun et al., 2018). DBP15k includes three cross-lingual datasets collected from DBpedia: Chinese and English (DBPZH-EN), Japanese and English (DBPJA-EN), French and English (DBPFR-EN). DWY100k contains two monolingual datasets: DBpedia and Wikidata (DBP-WD), DBpedia and YAGO (DBP-YG). The original DBP15k does not have attribute triples. Therefore we retrieve attribute triples from the DBpedia dump (2016-10). We then randomly sample 30% of gold entity alignments for training and use the rest for testing. For DWY100k, we use the released attribute triples and the train/valid/test split of Zhang et al. (2019). We show the number of relation/attribute triples for each dataset in Table 1.

Baselines. We compare AttrGNN with 12 baselines. We summarize four common characteristics of EA models and mark the employed characteristic for each method in Table 2. Among them, AttrE and MultiKE use the same information as AttrGNN. We also construct a baseline NameBERT that only uses the BERT embedding of entity names to measure the similarity. For each model, we list the reported performance if available; otherwise, we run the source code to get the result. Following ex-

Datasets	#Relation	#Digital	#Literal
DBPZH	153k	177k	290k
DBPEN	237k	203k	292k
DBPJA	164k	152k	227k
DBPEN	233k	171k	268k
DBPFR	192k	162k	313k
DBPEN	278k	227k	323k
DWYWD	463k	362k	628k
DWYDB	448k	219k	403k
DWYYG	428k	1147k	712k
DWYDB	502k	253k	506k

Table 1: Triple numbers of datasets. #Relation indicates the number of relation triples. The numbers of attribute triples that have digital values and literal values are denoted by #Digital and #Literal.

	Attr	Value	Name	Iter
MTransE (2017)				
JAPE (2017)	✓			
IPTransE (2017)				✓
AlignE (2018)				
BootEA (2018)				✓
KDCoE (2018)				✓
GCN-Align (2018)	✓			
MuGNN (2019a)				
AttrE (2019)	✓	✓	✓	
MultiKE (2019)	✓	✓	✓	
GraphMatch (2019)			✓	
RDGCN (2019a)			✓	
AttrGNN (Ours)	✓	✓	✓	

Table 2: Characteristics of entity alignment models. The top part lists 8 models without utilizing entity names, and the bottom part lists 5 models with entity names. Attr and Value indicate the attributes and values from attribute triples; Name indicates entity names; and Iter indicates whether the model iteratively enlarge training set of equivalent entities.

Methods	DBP$_{\text{ZH-EN}}$			DBP$_{\text{JA-EN}}$			DBP$_{\text{FR-EN}}$		
	H@1	**H@10**	**MRR**	**H@1**	**H@10**	**MRR**	**H@1**	**H@10**	**MRR**
MTransE	30.83	61.41	0.364	27.86	57.45	0.349	24.41	55.55	0.335
JAPE	41.18	74.46	0.490	36.25	68.50	0.476	32.39	66.68	0.430
AlignE	47.18	79.19	0.581	44.76	78.89	0.563	48.12	82.43	0.599
BootEA	62.94	84.75	0.703	62.23	85.39	0.701	65.30	87.44	0.731
GCN-Align	41.25	74.38	0.549	39.91	74.46	0.546	37.29	74.49	0.532
MuGNN	49.40	84.40	0.611	50.10	85.70	0.621	49.50	87.00	0.621
NameBERT	60.36	71.00	0.642	74.53	83.57	0.779	87.44	92.06	0.891
MultiKE*	43.70	51.62	0.466	57.00	64.26	0.596	71.43	76.08	0.733
GraphMatch	67.93	78.48	-	73.97	87.15	-	89.38	95.24	-
RDGCN	70.75	84.55	0.749*	76.74	89.54	0.812*	88.64	95.72	0.908*
AttrGNN$_{\text{avg}}$	**79.60**	**92.93**	**0.845**	**78.33**	**92.08**	**0.834**	91.85	97.77	0.910
AttrGNN$_{\text{svm}}$	77.72	92.00	0.829	76.25	90.88	0.816	**94.24**	**98.67**	**0.959**

Table 3: Overall performance on the regular setting of DBP15k. Models in the first part do not use *Names* while models in the second part use *Name*. * indicates results from our re-implementation using their source code.

isting works (Sun et al., 2018), we employ Hits@N (%, short as H@N) and Mean Reciprocal Rank (MRR) as the evaluation metrics. Higher Hits@N and MRR indicate better performance.

Training Details. We use BERT (Devlin et al., 2019) to initialize the feature vector for each value. Specifically, given a value v consisting of a sequence of tokens, we use the pre-trained *bert-base-cased*[3] to generate a sequence of hidden states and apply max-pooling to obtain a fixed length vector $\mathbf{v}$ as the initial value feature vector. We do not fine-tune the BERT so that the feature vectors can be cached for efficiency. Following Sun et al. (2017), we use Google Translate to translate all values to English for cross-lingual datasets. We initialize the four channels defined in Section 3.3 as follows. For the *Name* channel, we initialize the entity features using the BERT embedding of entity names. For the *Literal*, *Digital*, and *Structure* channels, we use randomly initialized the 128 dimensional vectors as the entity and attribute features. We use Adagrad (Duchi et al., 2011) as the optimizer. For each entity, we choose maximum 20 or 3 attribute triples based on GPU memory. For Graph Alignment, we choose 25 negative samples for each entity. We use 16 negative samples for each positive sample in the SVM ensemble model. We grid search the best parameters for each GNN channel on the valid set (if available) in the following range: learning rate $\{0.001, 0.004, 0.007\}$, L2 regularization $\{10^{-4}, 10^{-3}, 0\}$. We set $\gamma = 1.0$. We train each channel for 100 epochs. For the SVM in Channel Ensemble, we search for C in range $\{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$. The experiments are conducted on a server with two 6-core 2.40ghz CPUs, one TITAN X, and 128 GB memory. On DBP15k, the *Literal*/*Digital*/*Name* channel costs less than 20 minutes for a grid search, and *Structure* channel costs less than 5 minutes.

[3]https://github.com/huggingface/transformers

4.2 Overall Performance

We report the results in two settings: *regular setting*, *i.e.*, the setting used in the previous entity alignment works; and *hard setting*, where we construct a harder test set for objective evaluation.

4.2.1 Regular Setting

Cross-lingual Dataset. Table 3 shows the overall performance on DBP15K. We can see that:

1. As compared to the second best model, AttrGNN achieves significant performance improvements of 5.10% for Hits@1 and 0.056 for MRR on average. This demonstrates the effectiveness of AttrGNN in integrating both attribute triples and relation triples.

2. NameBERT, which only uses entity names, performs better than models without using names in most cases. This demonstrates our observations that (1) the datasets are name-biased; and (2) the evaluation result cannot reflect true EA performance in real-world situation. Specifically, NameBERT performs better on DBP$_{\text{FR-EN}}$ than that on DBP$_{\text{JA-EN}}$ and DBP$_{\text{ZH-EN}}$, which indicates a higher name-bias on DBP$_{\text{FR-EN}}$. The reason is the better translation quality between French and English.

3. AttrGNN's performance improvement over baselines is higher on DBP$_{\text{ZH-EN}}$ (8.85%) than those on DBP$_{\text{JA-EN}}$ (1.59%) and DBP$_{\text{FR-EN}}$ (4.86%). The primary reason is that on DBP$_{\text{ZH-EN}}$, different chan-

nels of features complement each other better than those on DBP$_{\text{JA-EN}}$ and DBP$_{\text{FR-EN}}$. The ratios[4] of complementary features on DBP$_{\text{ZH-EN}}$/DBP$_{\text{JA-EN}}$/DBP$_{\text{FR-EN}}$ are $19\%/9\%/5\%$. Thus, we benefit the most on DBP$_{\text{ZH-EN}}$ from the ensemble.

4. The SVM ensemble strategy performs better than average pooling on DBP$_{\text{FR-EN}}$. On DBP$_{\text{FR-EN}}$, the performances of AttrGNN channels are imbalanced: the *Name* channel performs much better than other channels, as shown by the performance gap between NameBERT and baselines without names on these datasets. In these imbalanced cases, SVM performs better because it can adjust the weights of channels. However, we can not explain that the SVM strategy performs worse that average pooling on DBP$_{\text{ZH-EN}}$ and DBP$_{\text{JA-EN}}$. In fact, the integration of the various KG features is an open problem. We leave that as a future work.

Monolingual Dataset. We evaluate models on this monolingual setting to inspect the name-bias level when there is no translation error. Table 4 shows the performance on DWY100K. The overall performance is similar to that on DBP15k, on which AttrGNN achieves the best performance. There are three major observations:

1. NameBERT achieves nearly 100% Hits@1 on DBP-YG, which shows more severe name-bias than that on the cross-lingual dataset. The reason is that both DBpedia and YAGO are derived from Wikipedia, resulting in that 77.60% of the released equivalent entities have exactly the same names while the rest have very similar names, *e.g.*, *George B. Rodney* and *George B Rodney*. This results dose not indicate that EA is solved because EA is still challenging when integrating KGs from different domains, where entity names can be very different.

2. AttrE and MultiKE, which use entity names, do not perform well because of their agnostic of attribute importance. The crucial alignment signal from *Name* is thus averaged away by other attribute triples (in DBpedia, each entity has 7-8 attribute triples in average).

3. MultiKE performs better than AttrE because it particularly sets a "Name View" to incorporate names. However, MultiKE performs worse than NameBERT on DBP-YG and DBP15k (Table 3), indicating that its inefficient combination of "Name View" and other views harms the performance.

[4]We test the ratios of two models', *i.e.*, the *Name* channel and the ensemble of the other three channels, complementary correct predictions.

Methods	DBP-WD			DBP-YG		
	H@1	H@10	MRR	H@1	H@10	MRR
MTransE	28.12	51.95	0.363	25.15	49.29	0.334
JAPE	31.84	58.88	0.411	23.57	48.41	0.320
IPTransE	34.85	63.84	0.447	29.74	55.76	0.386
BootEA	74.79	89.84	0.801	76.10	89.44	0.808
KDCoE	57.19	69.53	0.618	42.71	48.30	0.446
GCN-Align	47.70	75.96	0.577	60.05	84.14	0.686
MuGNN	61.60	89.70	0.714	74.10	93.70	0.810
NameBERT	83.32	90.15	0.860	99.85	99.99	0.999
AttrE	38.96	66.77	0.487	23.24	42.70	0.300
MultiKE	91.86	96.26	0.935	88.03	95.32	0.906
AttrGNN$_{\text{avg}}$	**96.08**	**98.86**	**0.972**	99.89	99.99	0.999
AttrGNN$_{\text{svm}}$	85.50	93.73	0.884	**99.96**	**100.00**	**1.000**

Table 4: Overall performance on DWY100K. The performance of AttrE is reported in Zhang et al. (2019).

4.2.2 Hard Setting

In the hard setting, we aim to carry out a more objective evaluation of EA models on a harder test set. We first introduce how to construct the test set and then present the results and discussion.

Build Harder Test Set. Let E_s and E'_s be the set of known aligned entities in G and G'. First, we compute the similarity matrix $\mathbf{S}$ via NameBERT; each element $\mathbf{S}_{e,e'}$ denotes the similarity between the entity pair $e \in E_s$ and $e' \in E'_s$. Second, we sort each row of $\mathbf{S}$ in descending order, by ranking (e, e') higher when there is less similarity in their names. Finally, we pick the highest-ranked 60% of equivalent entity pairs as the test set. The train set (30%) and the valid set (10%) are then randomly selected from the remaining set of data. We construct harder test set for the cross-lingual dataset only, because it is impractical to find equivalent entity pairs whose entities have very different names on the monolingual dataset, as shown by the performance of NameBERT in Table 4.

Discussion. We implement AttrGNN and eight best-performed baselines with their source codes on the hard setting. Table 5 shows the overall performance. We observe general performance drop in Hit@1 on DBP15k for all models, as shown in Figure 3. There are three major observations:

1. AttrGNN still achieves the best performance, demonstrating the effectiveness of our model. However, the performance of AttrGNN has degraded by around 6% for Hits@1. This degradation indicates that the practical application of EA is still challenging and worth exploration.

2. RDGCN shows the lowest degradation in performance among all the models with entity names

Methods	DBP$_{ZH-EN}$			DBP$_{JA-EN}$			DBP$_{FR-EN}$		
	H@1	**H@10**	**MRR**	**H@1**	**H@10**	**MRR**	**H@1**	**H@10**	**MRR**
JAPE	34.97	56.63	0.451	31.07	52.03	0.410	25.30	48.29	0.361
AlignE	40.09	69.94	0.501	37.42	69.19	0.479	38.01	71.28	0.492
BootEA	51.26	74.60	0.593	49.31	74.64	0.578	51.28	76.93	0.603
GCN-Align	36.59	64.66	0.464	33.94	65.30	0.448	30.32	63.69	0.414
MuGNN	40.64	74.58	0.521	39.86	75.33	0.515	40.71	78.26	0.531
NameBERT	38.36	55.06	0.444	60.03	74.47	0.654	79.02	86.89	0.820
MultiKE	27.92	35.21	0.306	48.18	55.68	0.509	64.69	69.54	0.665
GraphMatch	50.06	66.93	-	60.26	71.78	-	83.50	90.47	-
RDGCN	60.44	76.60	0.662	68.19	83.77	0.737	82.87	93.12	0.866
AttrGNN$_{avg}$	**66.21**	**81.81**	**0.719**	75.72	88.76	0.805	86.41	94.67	0.894
AttrGNN$_{svm}$	65.90	81.16	0.716	**77.39**	**90.33**	**0.821**	**88.64**	**95.64**	**0.912**

Table 5: Overall performance on the hard setting of DBP15k.

because RDGCN utilizes the feature of relation type within a GNN framework. This stable performance suggests that incorporating relation type into GNN is crucial for EA and worth exploration.

3. Except for the iterative model, *i.e.*, BootEA, the performance of models without using entity names exhibits less performance drop than the models with names. The iterative model's performance degrades more because the harder dataset weakens the snowball effect [5] when iteratively enlarging the seed set of equivalent entities.

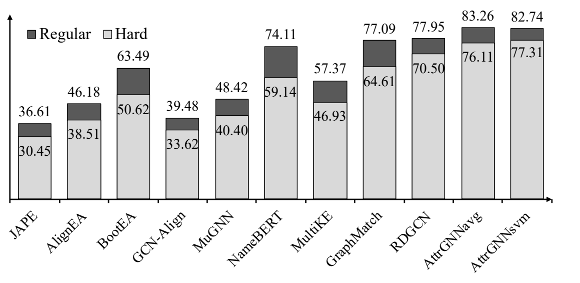


Figure 3: Average Hits@1 (%) of models under the regular setting and the hard setting on DBP15k.

4.3 Ablation Study

We conduct an ablation study on the performance of each AttrGNN channel, AttrGNN$_{avg}$ without using the *Name* channel (A w/o Name), AttrGNN without using relation triples (A w/o Relation), and AttrGNN without graph partition (MixAttrGNN) (Figure 4). A w/o Relation is to ensemble NameBERT and one-layer Literal and Digital channels. There are three major observations:

1. The *Literal* and *Structure* channels' performances are close to the *Name* channel under the hard setting. This demonstrates the importance to explore non-name features, including other attributes and relation, for practical EA.

2. Compared to MixAttrGNN, our simple graph partition strategy achieves promising improvement. The reason is that graph partition enables model to measure the similarity of different attributes differently.

3. The *Digital* channel's performance is poor because it is challenging to learn the numerical calculation with the supervision of entity alignment. We thus leave it as future work.

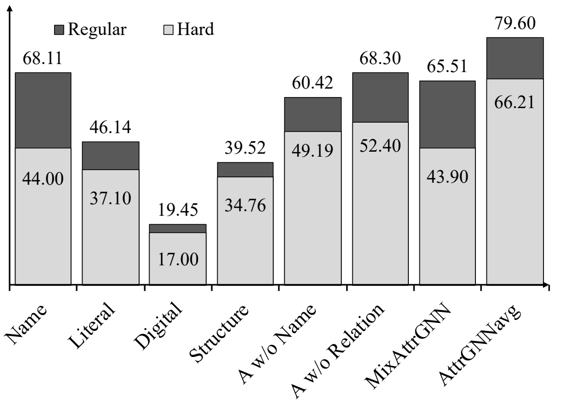


Figure 4: Ablation study on DBP$_{ZH-EN}$ (Hits@1 %).

4. Our full model significantly outperforms the Structure channel and the A w/o relation, which are the models with only relation/attribute features. This demonstrates the necessity of considering both relation and attribute triples for EA.

4.4 Case Study of Attributes and Values

We give a qualitative analysis of how attribute triples contribute to EA in this case study. Table 6 shows an equivalent entity pair that NameBERT fails to align, but AttrGNN aligns it by taking alignment signal from attributes and values. We observe

[5]https://en.wikipedia.org/wiki/Snowball_effect

Score	Attribute	Value
English Entity: Georgia (U.S. state)		
.109	postalabbreviation	GA
.039	former	Province of Georgia
.037	flag	Flag of Georgia.svg
.028	arearank	24
...		
.020	senators	David Perdue
.020	governor	Nathan Deal
.019	motto	Wisdom, Justice...
Chinese Entity: Georgia		
.144	postalabbreviation	GA
.048	flag	Flag of Georgia.svg
.041	fullZhName	Georgia
.037	arearank	24
...		
.026	officiallang	English
.026	admittancedate	1788
.025	totalarea	154077

Table 6: Attributes and values for the entity "Georgia (U.S. state)" from the English and Chinese DBpedia. Attributes are sorted in descending order according to the attention score. Chinese texts are translated.

that most of the top-ranked attributes have similar values between two KGs. In this case, the similar values include three literal strings, *e.g.*, *GA*, *Flag of Georgia* and *Seal of Georgia*, and a number, *e.g. 24*. Meanwhile, the values that are not shared in both KGs are assigned low attention weights and filtered out. As similar cases are commonly observed, we conclude that – attributes determine the importance of values, and values provide discriminative signals. In other words, the attributes whose values are unique are ranked higher, *e.g.*, *postalabbreviation* that denotes the unique postal abbreviation of provinces. The value of the lowest-ranked attributes may have different forms in different KGs. For example, the attention weight of *totalarea* is small, because English KG and Chinese KG use different units of area (square mile in English DBpedia and square kilometer in Chinese DBpedia).

5 Conclusion and Future Work

We propose a novel EA model (AttrGNN) and contribute a hard experimental setting for practical evaluation. AttrGNN can integrate both attribute and relation triples with varying importance for better performance. Experimental results under the regular and hard settings present significant improvements of our proposed model, and the severe dataset bias can be effectively alleviated in our proposed hard setting.

In the future, we are interested in replacing BERT with knowledge enhanced and number sensitive text representations models (Cao et al., 2017; Geva et al., 2020).

Acknowledgments

This research is supported by the National Research Foundation, Singapore under its International Research Centres in Singapore Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

References

Daniel Andor, Luheng He, Kenton Lee, and Emily Pitler. 2019. Giving bert a calculator: Finding operations and arguments with reading comprehension. In *EMNLP*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*.

Yixin Cao, Lei Hou, Juanzi Li, and Zhiyuan Liu. 2018. Neural collective entity linking. In *COLING*.

Yixin Cao, Lifu Huang, Heng Ji, Xu Chen, and Juanzi Li. 2017. Bridge text and knowledge by learning multi-prototype entity mention embedding. In *ACL*.

Yixin Cao, Zhiyuan Liu, Chengjiang Li, Juanzi Li, and Tat-Seng Chua. 2019a. Multi-channel graph neural network for entity alignment. In *ACL*.

Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. 2019b. Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *WWW*.

Muhao Chen, Yingtao Tian, Kai-Wei Chang, Steven Skiena, and Carlo Zaniolo. 2018. Co-training embeddings of knowledge graphs and entity descriptions for cross-lingual entity alignment. In *IJCAI*.

Muhao Chen, Yingtao Tian, Mohan Yang, and Carlo Zaniolo. 2017. Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. In *IJCAI*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*.

Michael R Garey and David S Johnson. 1990. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co.

Mor Geva, Ankit Gupta, and Jonathan Berant. 2020. Injecting numerical reasoning skills into language models. In *ACL*.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*.

Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

Shantanu Kumar. 2017. A survey of deep learning methods for relation extraction. *arXiv preprint*.

Chengjiang Li, Yixin Cao, Lei Hou, Jiaxin Shi, Juanzi Li, and Tat-Seng Chua. 2019. Semi-supervised entity alignment via joint knowledge embedding model and cross-graph model. In *EMNLP*.

Zequn Sun, Wei Hu, and Chengkai Li. 2017. Cross-lingual entity alignment via joint attribute-preserving embedding. In *ISWC*.

Zequn Sun, Wei Hu, Qingheng Zhang, and Yuzhong Qu. 2018. Bootstrapping entity alignment with knowledge graph embedding. In *IJCAI*.

Johan AK Suykens and Joos Vandewalle. 1999. Least squares support vector machine classifiers. *Neural processing letters*.

Bayu Distiawan Trisedya, Jianzhong Qi, and Rui Zhang. 2019. Entity alignment between knowledge graphs using attribute embeddings. In *AAAI*.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *ICLR*.

Zhichun Wang, Qingsong Lv, Xiaohan Lan, and Yu Zhang. 2018. Cross-lingual knowledge graph alignment via graph convolutional networks. In *EMNLP*.

Yuting Wu, Xiao Liu, Yansong Feng, Zheng Wang, Rui Yan, and Dongyan Zhao. 2019a. Relation-aware entity alignment for heterogeneous knowledge graphs. In *IJCAI*.

Yuting Wu, Xiao Liu, Yansong Feng, Zheng Wang, and Dongyan Zhao. 2019b. Jointly learning entity and relation representations for entity alignment. In *EMNLP*.

Yuting Wu, Xiao Liu, Yansong Feng, Zheng Wang, and Dongyan Zhao. 2020. Neighborhood matching network for entity alignment. In *ACL*.

Kun Xu, Liwei Wang, Mo Yu, Yansong Feng, Yan Song, Zhiguo Wang, and Dong Yu. 2019. Cross-lingual knowledge graph alignment via graph matching neural network. In *ACL*.

Junchi Yan, Xu-Cheng Yin, Weiyao Lin, Cheng Deng, Hongyuan Zha, and Xiaokang Yang. 2016. A short survey of recent advances in graph matching. In *ICMR*.

Shuo Yang, Lei Zou, Zhongyuan Wang, Jun Yan, and Ji-Rong Wen. 2017. Efficiently answering technical questions—a knowledge graph approach. In *AAAI*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*.

Qingheng Zhang, Zequn Sun, Wei Hu, Muhao Chen, Lingbing Guo, and Yuzhong Qu. 2019. Multi-view knowledge graph embedding for entity alignment. In *IJCAI*.

Hao Zhu, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2017. Iterative entity alignment via joint knowledge embeddings. In *IJCAI*.