Guiding Neural Entity Alignment with Compatibility

Bing Liu\textsuperscript{1,2}, Harrisen Scells\textsuperscript{1}, Wen Hua\textsuperscript{1}, Guido Zuccon\textsuperscript{1}, Genghong Zhao\textsuperscript{2}, Xia Zhang\textsuperscript{3}

\textsuperscript{1}The University of Queensland, Australia
\textsuperscript{2}Neusoft Research of Intelligent Healthcare Technology, Co. Ltd., China
\textsuperscript{3}Neusoft Corporation, China

\{bing.liu, h.scells, w.hua, g.zuccon\}@uq.edu.au
\{zhaogenghong, zhangx\}@neusoft.com

Abstract

Entity Alignment (EA) aims to find equivalent entities between two Knowledge Graphs (KGs). While numerous neural EA models have been devised, they are mainly learned using labelled data only. In this work, we argue that different entities within one KG should have compatible counterparts in the other KG due to the potential dependencies among the entities. Making compatible predictions thus should be one of the goals of training an EA model along with fitting the labelled data: this aspect however is neglected in current methods. To power neural EA models with compatibility, we devise a training framework by addressing three problems: (1) how to measure the compatibility of an EA model; (2) how to inject the property of being compatible into an EA model; (3) how to optimise parameters of the compatibility model. Extensive experiments on widely-used datasets demonstrate the advantages of integrating compatibility within EA models. In fact, state-of-the-art neural EA models trained within our framework using just 5% of the labelled data can achieve comparable effectiveness with supervised training using 20% of the labelled data.

1 Introduction

Knowledge Graphs (KGs) have been widely used across many Natural Language Processing applications (Ji et al., 2022). However, most KGs suffer from incompleteness which limits their impact on downstream applications. At the same time, different KGs often contain complementary knowledge. This makes fusing complementary KGs a promising solution for building a more comprehensive KG. Entity Alignment (EA), which identifies equivalent entities between two KGs, is essential for KG fusion. Given the two examples KGs shown in Fig. 1, EA aims to recognize two entity mappings Donald Trump $\equiv$ D.J. Trump and Fred Trump $\equiv$ Frederick Christ Trump.

Neural EA models are the current state-of-the-art for entity alignment (Sun et al., 2020b; Zhao et al., 2022; Zhang et al., 2020; Mao et al., 2021b) These methods use pre-aligned mappings to learn an EA model: it encodes entities into informative embeddings and then, for each source entity, selects the closest target entity in the vector space as its counterpart. Though significant progress has been achieved, the dependencies between entities, which is the nature of graph data, is underexplored. In an EA task, the counterparts of different entities within one KG should be compatible w.r.t. the underlying dependencies. For example, in Fig 1, the two mappings Donald Trump $\equiv$ D.J. Trump and Ivanka Trump $\equiv$ Melania Trump should not co-exist at the same time (i.e. they are incompatible) since ‘someone’s daughter and wife cannot be the same person’. On the contrary, Donald Trump $\equiv$ D.J. Trump and Fred Trump $\equiv$ Frederick Christ Trump are compatible mappings since equivalent entities’ father entities should also be equivalent. Through an experimental study, we verified that more effective EA models make more compatible predictions (see Appendix A for more details). Therefore, we argue that making compatible predictions should be one of the objectives of training a neural EA model, other than fitting the labelled data. Unfortunately, compatibility has thus far been neglected by the existing neural EA works.
To fill this gap, we propose a training framework EMEA, which exploits compatibility to improve existing neural EA models. Few critical problems make it challenging to drive a neural EA model with compatibility: (1) A first problem is how to measure the overall compatibility of all EA predictions. We notice some reasoning rules defined in traditional reasoning-based EA works (Suchanek et al., 2011) can reflect the dependencies between entities well. To inherit their merits, we devise a compatibility model which can reuse them. In this way, we contribute one mechanism of combining reasoning-based and neural EA methods. (2) The second problem is how to improve the compatibility of EA model. Compatibility is measured on the counterparts (i.e. labels) sampled from the EA model, but the sampling process is not differentiable and thus the popular approach of regularizing an item in the loss is infeasible. We overcome this problem with variational inference. (3) The third problem lies in optimising the compatibility model, which has interdependencies with the unknown counterparts. We solve this problem with a variational EM framework, which alternates updating the neural EA model and the compatibility model until convergence.

Our contributions can be summarized as:

- We investigate the compatibility issue of the neural EA model, which is critical but so far neglected by the existing neural EA works.
- We propose one generic framework, which can guide the training of neural EA models with compatibility apart from labelled data.
- Our framework bridges the gap between neural and reasoning-based EA methods.
- We empirically show compatibility is very powerful in improving neural EA models, especially when the training data is limited.

2 Related Work

Neural EA. Entity Alignment is an important task and has been widely studied. Neural EA (Sun et al., 2020b; Zhao et al., 2022; Zhang et al., 2020) is current mainstream direction which emerges with the development of deep learning techniques. Various neural architectures have been introduced to encode entities. Translation-based KG encoders were explored at the start (Chen et al., 2017; Zhu et al., 2017). Though these models could capture the structure information, they were not capable of incorporating attribute information. Graph Convolutional Network (GCN)-based encoders later became the mainstream method because they were flexible in combining different types of information and achieved higher performance (Wang et al., 2018; Cao et al., 2019; Mao et al., 2020a; Sun et al., 2020a; Mao et al., 2020b). Neural EA models rely on pre-aligned mappings for training (Liu et al., 2021a). To improve EA effectiveness, semi-supervised learning (self-training) was explored to generate pseudo mappings to enrich the training data (Sun et al., 2018; Mao et al., 2021b). Our work aims to complement the existing EA works regardless of their training methods.

Among previous neural EA methods, some were done on KGs with rich attributes and pay attention to exploiting extra information other than KG structure (Wu et al., 2019; Liu et al., 2021b, 2020; Mao et al., 2021c; Qi et al., 2021). Alternatively, some others only focused on designing novel models to extract better features from the KG structure (Sun et al., 2018, 2020a; Mao et al., 2020b; Liu et al., 2022) since structure is the most basic information and the proposed method would be more generic. We evaluate our method by applying it to models that only consider KG structure, which is a more challenging setting.

Reasoning-based EA. In the reasoning-based EA works (Saïs et al., 2007; Hogan et al., 2007; Suchanek et al., 2011), some rules are defined based on the dependencies between entities. With the rules, label-level reasoning, i.e. inferring the label of one entity according to other entities’ labels instead of its own features, was performed to detect more potential mappings from the pre-aligned ones. Functional relation (or attribute), which can only have one object for a certain subject, is critical for some reasoning rules (Hogan et al., 2007). One rule example is: $\exists r, e_1, e_1' : r(e_1, r, e_2), r(e_1', r, e_2'), r$ is functional, $e_1 \equiv e_1' \Rightarrow e_2 \equiv e_2'$. Saïs et al. proposed to combine multiple properties instead of only using a functional property since the combination of several weak properties can also be functional. Hogan et al. quantified functional as functionality in a statistical way. Suchanek et al. inherited the reasoning ideas from previous works and further transformed the logic rules into a probabilistic

---

1Our code and used data are released at https://github.com/uqbingliu/EMEA
form in their work named PARIS. Apart from (Suchanek et al., 2011), few recent works (Sun et al., 2020b; Zhao et al., 2022) verified PARIS can achieve promising performance. In this work, we reuse one reasoning rule defined in PARIS because it is very effective and representative.

Combining Reasoning-based and Neural EA. One previous work named PRASE (Qi et al., 2021) also explored the combination of neural and reasoning-based EA methods. It used a neural EA model to measure the similarities between entities and fed these similarities to the reasoning method of PARIS. Our work provides a different combination mechanism of these two lines of methods.

3 Notations & Problem Definition

Suppose we have two KGs $\mathcal{G}$ and $\mathcal{G}'$ with respective entity sets $E$ and $E'$. Each source entity $e \in E$ corresponds to one counterpart variable $y_e \in E'$. For simplicity, we denote the counterpart variables of a set $E$ of entities as $y_E$ collectively, while use $\hat{y}_e$ to represent an assignment of the unlabelled entities. $\mathcal{G}$ represents the parameters of the EA model.

4 The EMEA Framework

Fig. 2 shows an overview of our EMEA framework. Towards improving a given neural EA model, the EMEA performs the following core operations:

1. Normalises similarities between source entity $e$ and all target entities into distribution $q_\Theta(y_e)$; 
2. Measure the compatibility of all predictions by modelling the joint probability $p_\Phi(y_L, y_U)$ ($\Phi$ is parameters) of all (known or predicted) mappings; 
3. Derive more compatible predictions $q^*(y_U)$ (than current EA model) using the compatibility model to guide updating the EA model. 
4. To learn the parameters of the compatibility model, we devise one optimisation mechanism based on variational EM (Neal and Hinton, 1998), which alternates the update of $\Theta$ and $\Phi$. The neural EA model $\Theta$ is initially trained in its original way. In the M-step, we assume current $\Theta$ is correct, and sample $\hat{y}_U \sim q_\Theta$ to update $\Phi$. In the E-step, we in turn assume current $\Phi$ is correct, and exploit $p_\Phi$ to derive more compatible distribution $q^*(y_U)$. The neural EA model $\Theta$ is then updated using the samples $\hat{y}_U \sim q^*$ together with the labelled data. This EM process repeats until $\Theta$ converges.

4.1 Normalising EA Similarity to Probability

Our method relies on the distribution form of counterpart variable $y_e$ as will be seen. However, the existing neural EA models only output similarities. To solve this problem, we introduce a separate model to normalise the similarities into probabilities $^2$. Given entity $e \in E$ and similarities $s_\Theta(e, e')$, we use $s(e, e') = s_\Theta(e, e')$ and $d(e, e') = \max(s_\Theta(e, e')) - s_\Theta(e, e')$ as features of each target entity $e' \in E'$. These features are combined linearly and fed into a softmax function with a temperature factor $\tau$, as shown in Eq. 1 and 2. The parameters $\Omega = \{\omega_1, \omega_2, \omega_0, \tau\}$ are learned by minimizing cross-entropy loss on the labelled data, i.e. Eq. 3. With the obtained model, we can transform $s_\Theta(e, e')$ into $q_\Theta(y_e)$.

\[
f(e, e') = \omega_1 \cdot s(e, e') + \omega_2 \cdot d(e, e') + \omega_0 \quad (1)
\]

\[
Pr_\Omega(y_e = e') = \frac{\exp(f_\Theta(e, e')/\tau)}{\text{sum}\left(\exp(f_\Theta(e, \cdot)/\tau)\right)} \quad (2)
\]

\[
O_\Omega = - \sum_{e \in L} \log Pr_\Omega(y_e = \hat{y}_e) \quad (3)
\]

4.2 Measuring Compatibility

It is not easy to establish the distribution $p_\Phi(y_L, y_U)$ over a large number of variables $y_L, y_U$. To address this problem, we model $p_\Phi(y_L, y_U)$ with graphical model (Wainwright and Jordan, 2008). Some simple normalisation method like MinMax scaler were tried but led to poor results.
which can be represented by a product of local functions (i.e. local compatibility). Each local function only depends on a factor subset \( F \subset E \), which is small and can be checked easily.

### 4.2.1 Local Compatibility

One rule \( \kappa \) is defined on a set of labels \( y_F \) according to the potential dependencies among \( y_F \). The assignments of variables \( y_F \) meeting the rule \( \kappa \) are thought compatible. Given a rule set \( \mathcal{K} = \{\kappa_1, \kappa_2, \ldots, \kappa_{|\mathcal{K}|}\} \) and a factor subset \( F \) to check, we define the corresponding local compatibility (i.e. local function) as Eq. 4, where \( \Phi_\kappa(\cdot) \) is an indicator function, \( \Phi = \{\Phi_\kappa \in \mathcal{K}, \phi_0\} \) are the weights of rules.

\[
l(y_F) = \exp \left( \sum_{\kappa \in \mathcal{K}} \phi_\kappa \cdot \Phi_\kappa(y_F) + \phi_0 \right) \tag{4}
\]

Next, we use two concrete examples to explain local compatibility. The PARIS rule is the primary one used by our framework, while another is for exploring the generality of different rule sets.

**PARIS Rule** (Suchanek et al., 2011) can be understood intuitively as: one mapping \( y_e = e' \) can be inferred from (or supported by) the other mappings between their neighbours \( N_e \) and \( N_{e'} \).

Given the predicted mappings, we can build one factor subset \( F_e \) at each entity \( e \), which contains \( e \) and its neighbouring entities \( N_e \). Then, we check whether PARIS rule can be satisfied by mappings \( y_{F_e} \). For example, in Fig. 3, we want to check the PARIS compatibility at \( e_2 \). In plot (a), for the mapping \( y_2 = e'_2 \), we can find two mappings between the neighbours of \( e_2 \) and \( e'_2 \): \( y_1 = e'_1 \) and \( y_3 = e'_3 \). Also, they can provide supporting evidence for \( y_2 = e'_2 \); if two entities have equivalent father entities and equivalent friend entities, they might also be equivalent. However, in plot (b), \( y_2 = e'_4 \) cannot get this kind of supporting evidence since there is no mapping between the neighbours of \( e_2 \) and \( e'_4 \). Thus, the local PARIS compatibility at \( e_2 \) in plot (a) is higher than that in plot (b).

![Figure 3: Example of PARIS compatibility. In (a), \( y_2 = e'_2 \) can get supporting evidence from \( y_1 = e'_1 \) and \( y_3 = e'_3 \), while \( y_2 = e'_4 \) in (b) cannot. Thus, the compatibility at \( e_2 \) in (a) is higher than that in (b).](image)

**Rules for Avoiding Conflicts.** We notice that most neural EA works assume that there is no duplicate within one KG, and thus different entities should have different counterparts. Otherwise, EA model makes conflicting predictions for them.

To reduce alignment conflicts, at each entity \( e \), we build one factor subset \( F_e \), which includes \( e \) and its top-\( N \) nearest neighbours in the embedding space of EA model. Basically, \( y_e \) should follow the prediction of neural EA model, i.e. \( g_1(y_{F_e}) = \text{arg max}_{e' \in E} \Theta_\kappa(y_{F_e}, e') \). Further, \( y_e \) should be unique, i.e. \( g_2(y_{F_e}) = 1 \).

### 4.2.2 Overall Compatibility

We further formulate the overall compatibility by aggregating local compatibilities on all the factor subsets \( \mathcal{F} = \{F_e, e \in E\} \) as in Eq. 5, where \( z \) is for normalisation.

\[
p_\Phi(y_L, y_U) = \frac{1}{z} \prod_{F \in \mathcal{F}} l(y_F) \tag{5}
\]

\[
z = \sum_{y_U \in E^{[1]}} \prod_{F \in \mathcal{F}} l(y_F) \tag{6}
\]

Note that \( z \) is intractable because it involves integral over \( y_U \in E^{[1]} \), which is a very large space. For such computation reason, we avoid computing \( p_\Phi(y_L, y_U) \), conditional probability like \( p_\Phi(y_U | y_L) \), and marginal probability \( p_\Phi(y_L) \) directly in the following sections.

Instead, we will exploit \( p_\Phi(y_e | y_{-e}) \) (\( -e \) refers to \( E \setminus e \)), whose computation is actually much easier. As in Eq. 7, computing \( p_\Phi(y_e | y_{-e}) \) only involves a few factor subsets containing \( e \). MB\( e' \) is the Markov Blanket of \( e \), which only contains entities cooccurring in any factor subset with \( e \). See
Appendix B.1 for the derivation process.

\[ p_{\Phi}(y_{e}|y_{-e}) = \frac{\prod_{F|e \in F} l(y_{F})}{\sum_{c' \in E'} \prod_{F|e \in F} l(y_{F}|y_{e} = c')}, \]
\[ \approx p_{\Phi}(y_{e}|y_{\text{MB}^*}) \]  

(7)

### 4.3 Guiding Neural EA with Compatibility

Suppose we have \( p_{\Phi}(y_{L}, y_{U}) \) which can measure the compatibility well. We attempt to make the distribution \( q_{\Theta}(y_{U}) \) close to \( p_{\Phi}(y_{U}|y_{L}) \), so that variable \( y_{U} \) sampled from \( q_{\Theta} \) are compatible. To this end, we treat minimizing the KL-divergence \( \text{KL}(q_{\Theta}(y_{U})||p_{\Phi}(y_{U}|y_{L})) \) as one of the objectives of optimising neural EA model \( \Theta \).

However, it is difficult to minimize the KL-divergence directly. We solve this problem with variational inference (Hogan, 2002). As shown in Eq. 8 and 9, the KL-divergence can be written as the difference between observed evidence \( \log p_{\Phi}(y_{L}) \), which is irrelevant to \( \Theta \), and its Evidence Lower Bound (ELBO), i.e. Eq. 9 (see Appendix B.2 for derivation process.). Minimizing the KL divergence is equivalent to maximizing the ELBO, which is computationally simpler.

\[ \text{KL}(q_{\Theta}(y_{U})||p_{\Phi}(y_{U}|y_{L})) = \log p_{\Phi}(y_{L}) - \text{ELBO} \]
\[ \text{ELBO} = \mathbb{E}_{q_{\Theta}(y_{U})} \log p_{\Phi}(y_{U}, y_{L}) - \mathbb{E}_{q_{\Theta}(y_{U})} \log q_{\Theta}(y_{U}) \]  

(8)

Because \( y_{i} \) are independent in neural EA model, we have \( q(y_{U}) = \prod_{u \in U} q(y_{u}) \). In addition, we use pseudolikelihood (Besag, 1975) to approximate the joint probability \( p_{\Phi}(y_{U}, y_{L}) \) for simpler computation, as in Eq. 10. Then, ELBO can be approximated with Eq. 11 (see Appendix B.3 for derivation details), where \(-u\) denotes \( U \setminus u \).

\[ p_{\Phi}(y_{U}, y_{L}) = p_{\Phi}(y_{L}) \prod_{u \in U} p_{\Phi}(y_{u}|y_{1:u-1}, y_{L}) \]
\[ \approx p_{\Phi}(y_{L}) \prod_{u \in U} p_{\Phi}(y_{u}|y_{-u}) \]  

(10)

\[ O_{\Theta} = \sum_{u \in U} \mathbb{E}_{q_{\Theta}(y_{u})} \left[ \mathbb{E}_{q_{\Theta}(y_{-u})} \left[ \log p_{\Phi}(y_{u}|y_{-u}) \right] - \log q_{\Theta}(y_{u}) \right] \]  

(11)

Now, our goal becomes to maximize \( O_{\Theta} \) w.r.t. \( \Theta \). Our solution is to derive a local optima \( q^{*}(y_{U}) \) of \( q_{\Theta}(y_{U}) \) with coordinate ascent, and then exploit \( y_{U} \sim q^{*}(y_{U}) \) to update \( \Theta \). In particular, we initialize \( q^{*}(y_{U}) \) with current \( q_{\Theta}(y_{U}) \) firstly. Then, we update \( q^{*}(y_{u}) \) for each \( u \in U \) in turn iteratively. Everyday we only update a single (or a block of) \( q^{*}(y_{u}) \) with Eq. 12, which can be derived from \( \frac{\partial O_{\Theta}}{\partial q_{\Theta}(y_{u})} = 0 \) (see Appendix B.4 for derivation details), while keeping the other \( q^{*}(y_{-u}) \) fixed. This process ends until \( q^{*}(y_{U}) \) converges.

\[ q^{*}(y_{u}) \propto \exp \left( \mathbb{E}_{q_{\Theta}(y_{U})} \log p_{\Phi}(y_{u}|y_{\text{MB}^*}) \right) \]

(12)

Afterwards, we sample \( y_{u} \sim q^{*}(y_{u}) \) for \( u \in U \), and join \( y_{U} \) with the labelled data \( y_{L} \) to form the training data. Eventually, we update \( \Theta \) with the original training method of neural EA model.

### 4.4 Optimisation with Variational EM

Though we have derived a way of guiding the training of \( \Theta \) with \( p_{\Phi} \), it remains a problem to optimise the weights of rules \( \Phi \). Typically, we learn \( \Phi \) by maximizing the log-likelihood of observed data \( \log p_{\Phi}(y_{L}) \). However, as shown in Eq. 10, \( \log p_{\Phi}(y_{L}, y_{U}) \), \( \log p_{\Phi}(y_{L}) \) relies on the latent variables \( y_{U} \). We apply a variational EM framework (Neal and Hinton, 1998; Qu and Tang, 2019) to update \( \Theta \) and \( \Phi \) by turns iteratively. In E-step, we compute the expectation of \( \log p_{\Phi}(y_{L}, y_{U}) \), i.e. \( \mathbb{E}_{p_{\Phi}(y_{U}|y_{L})} \log p_{\Phi}(y_{L}, y_{U}) \). Here, we approximate \( p_{\Phi}(y_{U}|y_{L}) \) with \( q_{\Theta}(y_{U}) \) and use the pseudolikelihood to approximate \( p_{\Phi}(y_{L}, y_{U}) \); Accordingly, we obtain the objective Eq. 13 for optimization.

\[ O_{\Phi} = \mathbb{E}_{q_{\Theta}(y_{U})} \sum_{e \in E} \log p_{\Phi}(y_{e}|y_{-e}) \]  

(13)

### 4.5 Implementation

We take a few measures to simplify the computation. (1) In Eq. 12 and Eq. 13, it is costly to estimate distribution \( q^{*}(y_{u}) \) and \( p_{\Phi}(y_{u}) \) because \( y_{u} \)'s assignment space \( E' \) can be very large. Instead, we only estimate \( q^{*}(y_{u}) \) for the top \( K \) most likely candidates according to current \( q_{\Theta}(y_{u}) \). (2) Both Eq. 11 and Eq. 12 involve sampling from \( q_{\Theta}(y_{u}) \) for estimating the expectation. We only sample one \( y_{u} \) as in \( y_{u} = \arg \max_{c' \in E} q_{\Theta}(y_{u} = c') \) for
Algorithm 1: The EMEA Framework

1. Train neural EA model $\Theta$ using $\hat{y}_L$
2. Normalise EA similarity to get $q_{\Theta}(y_u)$
3. for iterations do
   // M-step
   4. Update $\Phi$ by maximizing Eq. 13
   // E-step
   5. Derive $q^*(y_U)$ with Eq. 12
   6. Sample $\hat{y}_u \sim q^*(y_u)$ for $u \in U$
   7. Update EA model $\Theta$ with data $\hat{y}_U \cup \hat{y}_L$
   8. Normalise EA similarity to get $q_{\Theta}(y_u)$

for each $u \in U$. (3) When computing $q^*(y_U)$ with coordinate ascent, we treat $U$ as a single block and update $q^*(y_U)$ for once.

We describe the whole process of EMEA in Alg. 1.

5 Experimental Settings

5.1 Datasets and Partitions

We choose five datasets widely used in previous EA research. Each dataset contains two KGs and a set of pre-aligned entity mappings. Three datasets are from DBP15K (Sun et al., 2017), which contains three cross-lingual KGs extracted from DBpedia: French-English ($fr_{en}$), Chinese-English ($zh_{en}$), and Japanese-English ($ja_{en}$). Each KG contains around 20K entities, among which 15K are pre-aligned. The other two datasets are from DWY100K (Sun et al., 2018), which consists of two mono-lingual datasets: dbp_yg extracted from DBpedia and Yago, and dbp_wd extracted from DBpedia and Wikidata. Each KG contains 100K entities which are all pre-aligned. Our experiment settings only consider the structural information of KGs and thus will not be affected by the problems of attributes like name bias in these datasets (Zhao et al., 2022; Liu et al., 2020).

Most existing EA works use 30% of the pre-aligned mappings as training data, which however was pointed out unrealistic in practice (Zhang et al., 2020). We explore the power of compatibility under different amounts of labelled data – 1%, 5%, 10%, 20%, and 30% of pre-aligned mappings, which are sampled randomly. Another 100 mappings are used as the validation set, while all the remaining mappings form the test set.

5.2 Metrics

EA methods typically output a ranked list of candidate counterparts for each entity. Therefore, we choose metrics for measuring the quality of ranking. We use Hit@1 (i.e., accuracy), Mean Reciprocal Rank (MRR) and Mean Rank (MR) to reflect the model performance at suggesting a single entity, a handful of entities, and many entities. Higher Hit@1, higher MRR, and lower MR indicate better performance. Statistical significance is performed using paired two-tailed t-test.

5.3 Comparable Methods

Baselines. We select baselines with the following considerations: (1) To examine the effect of compatibility, we compare EMEA with the original neural EA model. (2) We compare EMEA with PARIS, which performs reasoning with the rule, to gain insights on different ways of using the rules. (3) To compare different combination mechanisms of neural and reasoning-based EA methods, we add PRASE (Qi et al., 2021), which exploits neural models to improve PARIS (Suchanek et al., 2011), as one baseline.

Neural EA models. For our choice of neural models, we select RREA (Mao et al., 2020b), which is a SOTA neural EA model under both supervised (denoted as RREA (sup)) and semi-supervised (denoted as RREA (semi)) modes. In addition, we also choose Dual-AMN (Mao et al., 2021b), AliNet (Sun et al., 2020a) and IPTransE (Zhu et al., 2017) to verify the generality of EMEA across different neural models. These three neural models vary in performance (see Appendix C.1) and KG encoders.

Note direct comparison between EMEA and the existing neural EA methods is not fair. The EMEA is a training framework designed to enhance the existing neural EA models. Its effectiveness is reflected by the performance difference of neural EA models before and after being enhanced with EMEA. The details about reproducibility (e.g. hyperparameter settings, etc.) can be found in Appendix C.

6 Results

Comparison with Baselines In Table 1, we report the overall performance of EMEA with supervised RREA and the baselines. Note the results of PRASE and PARIS are much lower than those in the literature because we only use the structure
information of KGs for all the methods. We have the following findings:

(1) By comparing EMEA with RREA, we can see that EMEA can significantly improve RREA across all the datasets and percentages of labelled data, especially when the amount of labelled data is small. For instance, EMEA using 5% of labelled data can achieve comparable effectiveness with supervised RREA using 20% of labelled data.

(2) EMEA always outperforms PARIS with a big margin. Thus, EMEA provides a better way of using the same reasoning rule. PARIS can only do label-level inference based on the reasoning rule, while EMEA can combine the power of neural EA model and reasoning rule.

(3) Some existing works show PARIS have very competitive performance with the SOTA neural EA models when the attribute information can be used. However, we find its performance is actually much worse than the SOTA neural model RREA when only the KG structure is available. This is a complementary finding about PARIS to the literature.

(4) Regarding the combination of RREA and PARIS, EMEA outperforms PRASE across all datasets and annotation costs. Though PRASE can always improve PARIS, there are some cases where it is worse than only using RREA. The potential reason is PARIS becomes the bottleneck of PRASE. On the contrary, EMEA is more robust – it consistently performs better than separately running RREA or PARIS.

To conclude, EMEA can significantly improve the SOTA EA model RREA by introducing compatibility. It also provides an effective combination mechanism of neural and reasoning-based EA methods, which outperforms the existing methods.

The further explorations of EMEA are done on zh_en dataset if not specially clarified.

Generality across Neural EA Models To explore the generality of EMEA across neural EA models, we apply EMEA to another three models: Dual-AMN (Mao et al., 2021b), AliNet (Sun et al., 2020a) and IPTTransE (Zhu et al., 2017) other than RREA. We find that: (1) EMEA can bring improvements to the three EA models as shown in Fig.4 (a), (b) and (c). Also, no matter whether the neural model is better or worse than PARIS, their combination using EMEA is more effective than using

<table>
<thead>
<tr>
<th>Method</th>
<th>zh_en</th>
<th>fr_en</th>
<th>ja_en</th>
<th>zh_en</th>
<th>fr_en</th>
<th>ja_en</th>
</tr>
</thead>
<tbody>
<tr>
<td>RREA (sup)</td>
<td>0.720</td>
<td>0.797</td>
<td>16.7</td>
<td>0.742</td>
<td>0.821</td>
<td>9.2</td>
</tr>
<tr>
<td>RREA (semi)</td>
<td>0.720</td>
<td>0.797</td>
<td>16.7</td>
<td>0.742</td>
<td>0.821</td>
<td>9.2</td>
</tr>
<tr>
<td>PARIS</td>
<td>0.706</td>
<td>0.777</td>
<td>27.4</td>
<td>0.727</td>
<td>0.802</td>
<td>9.7</td>
</tr>
<tr>
<td>PRASE</td>
<td>0.706</td>
<td>0.777</td>
<td>27.4</td>
<td>0.727</td>
<td>0.802</td>
<td>9.7</td>
</tr>
</tbody>
</table>

Table 1: Overall performance of EMEA and PARSE in combining RREA (sup) and PARIS rule across different percentages (1%-30%) of annotations. Bold indicates best for the specific annotation percentage; all differences between RREA and other baselines are statistically significant ($p < 0.01$); the hyphen '-' means not applicable.

Table 2: Overall performance of EMEA in combining RREA (semi) with PARIS rule across different percentages (1%-30%) of annotations. Bold indicates best for the specific annotation percentage: all differences between RREA and EMEA are statistically significant ($p < 0.05$).
Impact of Compatibility on Training Process

In each iteration of the training procedure, the compatibility model derives more compatible predictions to assist in updating the neural EA model in the next iteration. To gain insights into their interaction, we examine the EA effectiveness of the two components in the training process. Fig. 5 shows this processes running on datasets with 1% and 5% annotated data. Step 0 is the original neural EA model. We make the following observations: (1) Overall, the effectiveness of the two components jointly increases across the training procedure. This phenomenon indicates that they indeed both benefit from each other. (2) The compatibility model performs better than the neural model in the early stage while worse later. Thus, we can conclude that the compatibility model can always boost the neural EA model, regardless of whether it can derive more accurate supervision signals.

Effect of Rule Sets on EMEA

To explore the effect of different rule sets on EMEA, we deploy another rule set used for avoiding conflicts (denoted as AvoidConf here) in EMEA. As shown in Fig. 6, AvoidConf can improve supervised RREA consistently, as well as semi-supervised RREA when the training data is of a small amount. However, it can only bring very slight (or even no) improvement to the semi-supervised RREA when the training data is > 10%. Also, its performance is always worse than PARIS rule. Therefore, PARIS rule is stronger in indicating the compatibility of EA than AvoidConf. This finding is intuitively reasonable because the PARIS rule can be used to infer new mappings, while AvoidConf can only indicate the potential errors.

Sensitivity to Hyper-parameters

We also analyse the sensitivity of parameter $K$ (discussed in Sec. 4.5) in EMEA under three annotation settings: 1%, 5% and 10%. Fig. 7 shows the performance of EMEA, measured with Hit@1 and MR, with respect to different $K$ values. We find that: (1) the
performance of EMEA fluctuates when $K < 10$ but is relatively stable when $K > 10$. (2) EMEA is more sensitive to $K$ in settings with fewer annotations. The reason is that a well-trained neural model can rank most true correspondences in the top positions and thus is less affected by the cut-off. (3) Large $K$ values (e.g., 50) only make EMEA slightly more effective (note lower MR is better). Since larger $K$ values require more computation, a small value outside the sensitive range is suggested, like 10 in Fig. 7.

8 Conclusion

Entity Alignment is a primary step of fusing different Knowledge Graphs. Many neural EA models have been explored and achieved SOTA performance. Though, one nature of graph data – the dependencies between entities – is under-explored. In this work, we raise attention to one neglected aspect of EA task – different entities have compatible counterparts w.r.t. their underlying dependencies. We argue that making self-consistent predictions should be one objective of training EA model other than fitting the labelled data. Towards this goal, we devise one training framework named EMEA, which can intervene in the update of EA model to improve its compatibility. In EMEA, we address three key problems: (1) measure compatibility with a graphical model which can aggregate local compatibilities; (2) guide the training of EA models with the compatibility measure via variational inference; (3) optimize the compatibility module with a variational EM framework. We empirically show that compatibility is very powerful in driving the training of neural EA models. The EMEA complements the existing neural EA works.

Limitations

Our EMEA framework has a higher computation cost than the original neural EA model. It needs to compute the compatibility module and continue updating the neural EA model in the iterations. As a result, more time is taken to train the neural EA model. In addition, we only measure compatibility in one direction (i.e. selecting one KG as the source KG and measuring compatibility on the target KG). Considering both directions might be able to further increase the EA performance.

In future, we plan to explore dual compatibility modules. Also, we will study how to combine compatibility and self-training.

Acknowledgements

This research is supported by the National Key Research and Development Program of China No. 2020AAA0109400, the Shenyang Science and Technology Plan Fund (No. 21-102-0-09), and the Australian Research Council (No. DE210100160 and DP200103650).

References


Aidan Hogan, Axel Polleres, Jürgen Umbrich, and Antoine Zimmermann. 2010. Some entities are more equal than others: statistical methods to consolidate linked data. In 4th International Workshop on New Forms of Reasoning for the Semantic Web: Scalable and Dynamic (NeFoRS2010).


A Preliminary Study

Our preliminary study examines the compatibilities of neural EA models with different performances. We choose RREA (Mao et al., 2020b) as the neural EA model and train it with different amounts of labelled data. As for the compatibility, we measure it with PARIS rule and the number of conflicting predictions (see Sec. 4.2). The PARIS compatibility is shown in Fig. 8, where the violins represent the distributions of compatibility scores while the triangles represent the average compatibilities. We can observe that RREA always makes numerous incompatible predictions, especially when trained with few annotated entities, while most oracle alignments are compatible. The curve strongly suggests that a better model makes more compatible predictions. Similar findings can be observed in Fig. 8 (b), where fewer conflicts mean better compatibility. These observations motivate us to improve the neural EA methods by guiding them to better compatibility.

B Derivation Processes

B.1 Simplification of $p_{\Phi}(y_{e}|y_{-e})$

The derivation process of Eq. 7 is shown in Eq. 14. MB$^e$ is the Markov Blanket of $e$ and only contains entities cooccurring in any factor subset with $e$. We only need to sample $y_{MB^e}$ to compute $p_{\Phi}(y_{e}|y_{-e})$.

B.2 Evidence Lower Bound

The relationship between KL divergence and ELBO shown in Eq. 8 and Eq. 9 can be derived through Eq. 15.
We write $Q_{\Theta}$ in the form of Eq. 17, and then differentiate $Q_{\Theta}$ regarding $q_{\Theta}(y_e)$ as in Eq. 18. By letting $\frac{dQ_{\Theta}}{dy_e} = 0$, we can get Eq. 12. Note that the obtained $q^*$ in Eq. 12 needs to be normalized into probabilities since it is proportional to (i.e. $\propto$) the right size, which is not normalized.

B.5 Indicator Function of PARIS Rule

With symbols used in this work, the probabilistic form of PARIS rule, i.e. our indicator function, can be written as Eq. 19, where $r(e,n)$ denote any triple with $e$ as head entity, $r'(y_e,n')$ denote any triple with $y_e$ as head entity. In addition, $Pr(r' \subseteq r)$ represents the likelihood that $r'$ is a subrelation of $r$ ($Pr(r \subseteq r')$ is analogous), while $fun^{-1}(r)$ denotes the reverse functionality of relation $r$. See (Suchanek et al., 2011) for more details about them.

$$g(y_F) = Pr(e \equiv y_e) = 1 - \prod_{r(e,n),r'(y_e,n')} \left(1 - Pr(r' \subseteq r) \times fun^{-1}(r') \times q_{\Theta}(y_n = n') \right) \times \left(1 - Pr(r \subseteq r') \times fun^{-1}(r') \times q_{\Theta}(y_n = n') \right)$$  (19)

C. Experiments

C.1 Performance of Neural EA Models

Table 3 summarizes the performance of SOTA neural EA models. The results of IPTransE, GCN-Align, MUGNN, RSN, AliNet are reported in (Sun et al., 2020a) while others are reported in their original papers. The results of RREA trained with 30% of training data in Table 1 are reproduced by us and slightly different from the results reported in Table 3 because of different random settings. Similar situation can be found in Table 2.

C.2 Details for Reproducibility

Hyper-parameters We search the number of candidate counterparts $K$ from [5,10,20,50], and set it as 10 for the trade-off of effectiveness and efficiency as discussed in Sec. 7. The number of nearest neighbours $N$ in the AvoidConf rule is searched from [5,10,15,20] and set as 5 for similar consideration as $K$;

Implementation of Baselines and Neural EA Models The source codes of PARIS 4 and PRASE 5 are used to produce their results. As for the neural EA models, RREA 6 and Dual-AMN 7 are implemented based on their source codes, while AliNet and IPTransE are implemented with OpenEA 8. We use the default settings of their hyper-parameters in these source codes.

Configuration of Running Device We run the experiments on one GPU server, which is configured with an Intel(R) Xeon(R) Gold 6128 3.40GHz CPU, 128GB memory, 3 NVIDIA GeForce GTX 2080Ti GPU and Ubuntu 20.04 OS.

C.3 Running Time

We report our running time on datasets zh_en (15K) and dbp_wd (100K) in Table 4 and 5 as refer-
Table 3: Performance of neural EA baselines. Percentage of labelled data: 30%; \textit{sup}: supervised; \textit{semi}: semi-supervised.

<table>
<thead>
<tr>
<th>Method</th>
<th>zh_en MRR</th>
<th>ja_en MRR</th>
<th>fr_en MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hit@1</td>
<td>MRR</td>
<td>Hit@1</td>
</tr>
<tr>
<td>IPTransE (Zhu et al., 2017)</td>
<td>0.406</td>
<td>0.516</td>
<td>0.367</td>
</tr>
<tr>
<td>GCN-Align (Wang et al., 2018)</td>
<td>0.413</td>
<td>0.549</td>
<td>0.399</td>
</tr>
<tr>
<td>MuGNN (Cao et al., 2019)</td>
<td>0.494</td>
<td>0.611</td>
<td>0.501</td>
</tr>
<tr>
<td>RSN (Guo et al., 2019)</td>
<td>0.508</td>
<td>0.591</td>
<td>0.507</td>
</tr>
<tr>
<td>AliNet (Sun et al., 2020a)</td>
<td>0.539</td>
<td>0.628</td>
<td>0.549</td>
</tr>
<tr>
<td>MRAEA (sup) (Mao et al., 2020a)</td>
<td>0.638</td>
<td>0.736</td>
<td>0.646</td>
</tr>
<tr>
<td>PSR (sup) (Mao et al., 2021a)</td>
<td>0.702</td>
<td>0.781</td>
<td>0.698</td>
</tr>
<tr>
<td>RREA (sup) (Mao et al., 2020b)</td>
<td>0.715</td>
<td>0.794</td>
<td>0.713</td>
</tr>
<tr>
<td>Dual-AMN (sup) (Mao et al., 2021b)</td>
<td>0.731</td>
<td>0.799</td>
<td>0.726</td>
</tr>
<tr>
<td>BootEA (semi) (Sun et al., 2018)</td>
<td>0.629</td>
<td>0.703</td>
<td>0.622</td>
</tr>
<tr>
<td>MRAEA (semi) (Mao et al., 2020a)</td>
<td>0.757</td>
<td>0.827</td>
<td>0.758</td>
</tr>
<tr>
<td>PSR (semi) (Mao et al., 2021a)</td>
<td>0.802</td>
<td>0.851</td>
<td>0.803</td>
</tr>
<tr>
<td>RREA (semi) (Mao et al., 2020b)</td>
<td>0.801</td>
<td>0.857</td>
<td>0.802</td>
</tr>
<tr>
<td>Dual-AMN (semi) (Mao et al., 2021b)</td>
<td>0.808</td>
<td>0.857</td>
<td>0.801</td>
</tr>
</tbody>
</table>
Table 4: Time consumption (seconds) of running EMEA on zh_en.

<table>
<thead>
<tr>
<th>Anno.</th>
<th>Initialization</th>
<th>Neural Module</th>
<th>Joint Distr. Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 %</td>
<td>254.1</td>
<td>108.1±1.2</td>
<td>88.5±1.1</td>
</tr>
<tr>
<td>5 %</td>
<td>249.7</td>
<td>108.0±2.5</td>
<td>98.2±15.5</td>
</tr>
<tr>
<td>10 %</td>
<td>239.2</td>
<td>108.6±0.9</td>
<td>86.0±0.3</td>
</tr>
<tr>
<td>20 %</td>
<td>252.8</td>
<td>109.6±1.6</td>
<td>86.7±0.6</td>
</tr>
<tr>
<td>30 %</td>
<td>252.4</td>
<td>111.1±2.0</td>
<td>90.2±2.7</td>
</tr>
</tbody>
</table>

Table 5: Time consumption (minutes) of running EMEA on dbp_wd.

<table>
<thead>
<tr>
<th>Anno.</th>
<th>Initialization</th>
<th>Neural Module</th>
<th>Joint Distr. Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 %</td>
<td>169.1</td>
<td>36.8±0.6</td>
<td>19.1±1.4</td>
</tr>
<tr>
<td>5 %</td>
<td>172.2</td>
<td>33.0±6.6</td>
<td>15.9±3.9</td>
</tr>
<tr>
<td>10 %</td>
<td>146.3</td>
<td>43.5±3.2</td>
<td>22.8±2.4</td>
</tr>
<tr>
<td>20 %</td>
<td>190.0</td>
<td>45.3±3.3</td>
<td>29.2±2.0</td>
</tr>
<tr>
<td>30 %</td>
<td>183.9</td>
<td>49.0±4.4</td>
<td>30.2±6.9</td>
</tr>
</tbody>
</table>

ence. The experiments on other datasets take comparable running time as them w.r.t. corresponding dataset size. Note that these experiments are run on a shared server and cannot be used to measure the precise running efficiency. In each table, we count the time consumption of initializing EA model, updating EA model and computing the self-consistency module in each EM iteration. The experiments on dbp_wd (100K) are slow because the source code of RREA can only run on CPU while raising Out-of-Memory exception on GPU.