Dependency-aware Self-training for Entity Alignment

Bing Liu
bing.liu@uq.edu.au
The University of Queensland

Wen Hua
w.hua@uq.edu.au
The University of Queensland

Tiancheng Lan
tiancheng.lan@uqconnect.edu.au
The University of Queensland

Guido Zuccon
g.zuccon@uq.edu.au
The University of Queensland

ABSTRACT
Entity Alignment (EA), which aims to detect entity mappings (i.e. equivalent entity pairs) in different Knowledge Graphs (KGs), is critical for KG fusion. Neural EA methods dominate current EA research but still suffer from their reliance on labelled mappings. To solve this problem, a few works have explored boosting the training of EA models with self-training, which adds confidently predicted mappings into the training data iteratively. Though the effectiveness of self-training can be glimpsed in some specific settings, we still have very limited knowledge about it. One reason is the existing works concentrate on devising EA models and only treat self-training as an auxiliary tool. To fill this knowledge gap, we change the perspective to self-training to shed light on it. In addition, the existing self-training strategies have limited impact because they introduce either much False Positive noise or a low quantity of True Positive pseudo mappings. To improve self-training for EA, we propose exploiting the dependencies between entities, a particularity of EA, to suppress the noise without hurting the recall of True Positive mappings. Through extensive experiments, we show that the introduction of dependency makes the self-training strategy for EA reach a new level. The value of self-training in alleviating the reliance on annotation is actually much higher than what has been realised. Furthermore, we suggest future study on smart data annotation to break the ceiling of EA performance.

CCS CONCEPTS
• Information systems → Information integration.

KEYWORDS
Knowledge Graph, Entity Alignment, Self-training, Dependency

1 INTRODUCTION
Knowledge Graphs (KGs), which represent entities and their relationships with graphs, have been widely used as knowledge drivers for many applications, such as semantic search, entity search, and recommender systems [9, 18, 18]. Due to the confined knowledge source and imperfect knowledge extraction techniques, a KG can hardly avoid the incompleteness issue. To solve this problem, fusing different KGs into a more comprehensive one is thought a promising direction. Entity Alignment (EA), which identifies equivalent entities between two KGs, is a critical step in KG fusion. Neural models dominate current EA research. The general idea of these methods is: encode the entities from two different KGs into the same embedding space; then, for each source entity, the target entity with the highest similarity in the embedding space is selected as its counterpart. Effective training is critical for getting informative entity embeddings. Supervised training relies on some pre-aligned mappings (i.e. seed mappings) as training data, which are costly to annotate (as in Fig. 1 (a)). To improve EA effectiveness without increasing annotation cost, self-training, a category of semi-supervised learning, has been explored to train the EA models more effectively [16, 21, 32]. In self-training strategies, we first train the EA model with labelled data. Then, we select some predicted mappings with high confidence (i.e. pseudo mappings) to add to the training data (together with the labelled data) and update/re-train the EA model (as in Fig. 1 (b)). Basically, the predicted mappings with high similarities are picked up [32]. To further improve the accuracy of pseudo mappings, Sun et al. [21] impose a one-to-one constraint on the derivation process; Alternatively, Mao et al. [16] choose to use two entities (from both KGs) which are mutually nearest neighbours in the embedding space to form a pseudo mapping. These works demonstrate that their proposed self-training strategies can significantly boost their specific EA models than
supervised learning. Though promising, the existing works about self-training for EA have apparent limitations.

Self-training is still underexplored and has never been studied systematically in EA task. The existing works [16, 32] mainly focus on designing novel EA models, while treat self-training as an auxiliary tool for improving their model performance. In other words, most attention was on the EA model but not the self-training strategies. As a result, self-training for EA has not been well investigated, and a series of questions remain to be answered. For example, which self-training method is the most effective? Are those self-training approaches generic across different EA models? To what extent can self-training alleviate the reliance of EA models on labelled data? and etc. To fill this gap, we benchmark the existing self-training methods for EA to shed light on them.

The existing self-training strategies for EA are based on the confidence of the EA model, i.e. similarities of predicted mappings. They can easily introduce False Positive noise – the pseudo mappings the model has high confidence in but are actually incorrect. This kind of noise can hardly be avoided, especially when the model cannot be well-trained, such as in the few annotation settings. As a result, the noise can propagate errors to the model in the next iteration of training and exaggerate the wrong patterns learned in previous iterations. Unfortunately, the existing measures used to control the noise, like increasing the threshold of selecting pseudo mappings, usually come with the cost of reduced True Positive pseudo mappings, which impairs the impact of self-training in another way. To escape such a dilemma, we believe the heuristic for generating pseudo mappings should go beyond the confidence of EA model.

We claim that exploiting the potential dependencies between entities, which is a particularity of graph data, is promising in improving the self-training strategy for EA. In EA task, this dependency can be interpreted as: the counterpart of one entity can affect the counterparts of its neighbouring entities. Because neural EA models perform inference independently, it is very possible for them to make predictions that violate the dependencies between entities. In our terminology, a set of mappings without violating any dependency are compatible, otherwise incompatible. Intuitively, a set of better predictions should be more compatible. Thus, by checking the violation of dependencies within a set of predictions, we have the chance to detect the suspicious ones and derive more compatible predictions. This would be helpful for generating less noisy pseudo mappings in self-training, as shown in Fig. 1 (c).

In this work, we propose a Self-Training framework for EA named STEA, which can incorporate the dependencies between entities to improve the effectiveness of training EA models. In our STEA framework, the EA model is initially trained with labelled data. Then, given a set of similarity-based EA predictions, we measure their overall compatibility using a probabilistic graphical model, and adjust them to be more compatible. For each entity, we derive a dependency-aware probability distribution over all counterpart candidates. The likelihood of candidate causing low compatibility will be suppressed. Finally, we generate pseudo mappings based on the dependency-aware EA predictions. A pair of entities having the highest probabilities on each other mutually form one pseudo mapping. The obtained pseudo mappings are added to the training data to update the EA model in conjunction with the labelled data. This process iterates until the EA model converges.

Our contributions can be summarised as below:

- We provide a systematic study of existing self-training methods for EA through benchmarking.
- We propose a self-training framework which can exploit the dependencies between entities to overcome the drawbacks of model confidence-based strategies.
- Extensive experiments have verified the effectiveness of our STEA framework. We find dependency can be used to improve the self-training strategy for EA task significantly.
- We find the impact of self-training on EA models is substantially undervalued. Our STEA framework can greatly alleviate the reliance of EA models on annotations.

2 RELATED WORK

Neural EA Model. Neural EA models [10, 23, 30, 31] emerge with the development of deep learning techniques and has been the mainstream of current EA research. KG encoders lie in the core of neural EA models. Various neural architectures have been explored to encode entities. The initial works tried translation-based models [6, 32]. Later, the emergence of Graph Convolutional Network (GCN) [28] changed the landscape of neural EA models. Many GCN-based KG encoders for EA task were designed and showed significant superiority than the translation-based models [16, 17, 22, 25]. Specifically, the GCN-based EA models can unify the way of encoding structure and attribute information naturally, while it is not easy for the translation-based models to exploit the attributes in KGs. In addition, the GCN-based methods can achieve state-of-the-art (SOTA) performance with big leads. In this work, we choose different GCN-based SOTA EA models to study the training effectiveness of different self-training frameworks on them. Among previous neural EA works, some only considered structure information of KG and dedicated to the modelling of KG structure [17, 21, 22], while some others focus on exploiting extra information in KGs other than the structure [12, 13, 27]. The former setting is more challenging and more general since structure is the most basic information in KGs. We follow this setting in our experiments.

Semi-supervised Learning for EA. Semi-supervised learning is an approach to machine learning that combines a small amount of labelled data with a large amount of unlabelled data during training. Self-training is a wrapper method for semi-supervised learning [1]. In general, the model is initially trained with labelled data. Then, it is used to assign pseudo-labels to the set of unlabelled training samples and enrich the training data and train a new model in conjunction with the labelled training set. In EA research, different self-training strategies have been explored. Zhu et al. [32] added all predicted mappings into training data while assigned high confidence predictions high weights in their training loss. Sun et al. [21] improved the self-training strategy further by imposing a one-to-one constraint in generating pseudo mappings. Mao et al. [16] proposed a strategy based on the asymmetric nature of alignment directions. If and only if two entities are mutually nearest neighbours, they form one pseudo mapping. Compared with the above two methods, this strategy does not introduce any hyperparameter and is very simple in implementation.

Our code, used data, and running scripts are released at https://github.com/uqbingliu/STEA.
Apart from self-training, co-training is another semi-supervised method explored to improve the effectiveness of EA models. Co-training is an extension of self-training in which multiple models are trained on different sets of features and generate labelled examples for one another. In EA task, Chen et al. [5] performed co-training of two EA models taking different types of KG information as input. Xin et al. [29] extended the number of EA models to \( k > 2 \).

Our work focuses on improving self-training for EA by producing better pseudo mappings based on entity dependency in the KG.

### 3 NOTATIONS & PROBLEM DEFINITION

Suppose we have two KGs \( G \) and \( G' \) with their corresponding entity sets \( E \) and \( E' \). Given \( G \), \( G' \) and a set of labelled entity mappings (i.e. seed mappings) \( M \equiv \{(e \in E, e' \in E')\} \), EA aims to identify more potential mappings.

For the convenience of explaining our method, suppose we treat \( G \) as the source KG while \( G' \) as the target KG. We denote the counterpart variable of each source entity \( e \in E \) as \( y_e \in E' \), while represent its assignment as \( y_e \). For simplicity, we denote the counterpart variables of a set \( E \) of entities as \( y_E \) collectively, i.e. \( y_E \equiv \{y_e | e \in E\} \). The sets of labelled and unlabelled source entities are denoted as \( L \subseteq E \) and \( U \subseteq E \) respectively.

In a neural EA method, one model \( \Theta \) can measure the similarity \( \text{Sim}_\Theta(e, e') \) between each source entity \( e \in E \) and each target entity \( e' \in E' \). The counterpart of entity \( e \) can be inferred via \( \hat{y}_e = \text{arg max}_{e' \in E'} \text{Sim}_\Theta(e, e') \).

### 4 THE STEA FRAMEWORK

Fig. 2 shows an overview of our STEA framework. In each iteration, STEA performs the following operations: 1. Train the neural EA model with the training mappings, which are the labelled mappings initially and will incorporate the pseudo mappings since the second iteration; 2. Normalise the similarities \( \{\text{Sim}_\Theta(e, e'), \forall e' \in E'\} \) between each source entity \( e \) and all target entities into probability distribution \( q(y_e) = \{\text{Pr}(y_e = e'), \forall e' \in E'\} \). For this purpose, a normalisation model is learned separately (Sec. 4.1); 3. Model the overall compatibility of current predictions \( \{q(y_u), u \in U\} \) by modelling the joint probability \( p(y_u, y_U) \) of all labelled and predicted mappings. In particular, we compute the local compatibility at each entity firstly and then aggregate all local compatibilities into a global one (Sec. 4.2); 4. Derive dependency-aware (i.e. more compatible) predictions \( \{q'(y_u), u \in U\} \) from \( \{q(y_u), u \in U\} \) with the assistance of the compatibility model (Sec. 4.3). This step is designed to suppress the likelihood of suspicious predictions. Since either KG can be treated as the source KG while the other one is seen as the target KG, we can also derive \( \{q'(y_u), u' \in U'\} \) from \( G' \) in the same way; 5. Generate pseudo mappings and combine them with the labelled mappings to enrich the training data. Based on the dependency-aware predictions, we explore a few different strategies for producing the pseudo mappings (Sec. 4.4). The iteration process (i.e. 1-5) repeats until \( \Theta \) converges.

#### 4.1 Normalising Similarity to Probability

Our method relies on the probabilities \( q(y_u) = \{\text{Pr}(y_u = e'), \forall e' \in E'\} \) of counterpart candidates for each source entity \( e \), while the existing neural EA models only output its similarities \( \{\text{Sim}(e, e'), \forall e' \in E'\} \) with all candidates. Therefore, we need to normalise the similarities into probabilities.
Based on this intuition, Suchanek et al. [19] designed a reasoning-based EA method named PARIS, which could achieve promising EA performance [23, 31]. In this work, we adapt the reasoning technique proposed in PARIS [19] to quantify the compatibility, as formulated in Eq. 4. Here, \( r_e \) denotes the entity set containing \( e \) and its neighbours \( N_e \), and is called a factor subset; \( r (e, n) \) represents a certain triple with head entity \( e \), relation \( r \), and tail entity \( n \in N_e \); \( Pr (r' \subseteq r) \) denotes the probability that \( r' \) is a sub-relation of \( r \), while \( \text{fun}^{-1} (r) \) denotes the inverse functionality of relation \( r \) (see [19] for more detailed explanation). The item \( y_n = y_n' \) indicates whether \( y_n \) equals to \( y_n' \). The whole equation of \( g(y_e = e') \) expresses the likelihood that prediction \( y_e \) can be inferred from its neighbouring predictions \( y_N_e \).

\[
g(y_{F_e}) = 1 - \prod_{r (e, n), r' (y_e, y_n')} \left( 1 - \Pr (r' \subseteq r) \times \text{fun}^{-1} (r) \times 1_{y_n = y_n'} \right) \times \left( 1 - \Pr (r' \subseteq r') \times \text{fun}^{-1} (r') \times 1_{y_n = y_n'} \right)
\]

(4)

We emphasize that: (1) When checking the local compatibility at entity \( e \), only a few predictions \( y_{F_e} \) are involved; (2) The compatibility score \( g(y_{F_e}) \) at entity \( e \) does not mean the correctness of prediction \( y_e \) exactly. High compatibility is a good signal indicating \( y_e \) is a correct prediction, but a low compatibility score might be caused by wrong \( y_e \) or wrong neighbouring predictions \( y_N_e \).

4.2.2 Overall Compatibility. We further formulate the overall compatibility by aggregating local compatibilities at all entities \( e \in E \) as in Eq. 5, where \( z \) is for normalisation.

\[
p(y_L, y_U) = \frac{1}{z} \prod_{e \in E} \exp \left( g(y_{F_e}) \right), \quad z = \prod_{y_U \in E^{[U]} \setminus E} \prod_{e \in E} \exp \left( g(y_{F_e}) \right)
\]

(5)

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

4.3 Deriving Dependency-aware Predictions

Precise predictions should have high overall compatibility. To this end, we try to update \( g(y_U) \) so as to minimise the KL-divergence \( \text{KL}(q(y_U) || p(y_U | y_L)) \). However, it is difficult to minimise the KL-divergence directly since we even cannot derive \( p(y_U | y_L) \) as mentioned in Sec. 4.2.2. To overcome this problem, we apply variational inference [8], in which we optimise the Evidence Lower Bound (ELBO) of the KL-divergence. As shown in Eq. 6 and 7, the KL-divergence can be written as the difference between observed evidence log \( p(y_L) \) and its ELBO. Thus, minimising the KL-divergence is equivalent to maximising the ELBO, which is computationally simpler as will be seen below.

\[
\text{KL}(q(y_U) || p(y_U | y_L)) = \log p(y_L) - \text{ELBO}\n\]

(6)

\[
\text{ELBO} = \mathbb{E}_{q(y_U)} \log p(y_U | y_L) - \mathbb{E}_{q(y_U)} \log q(y_U) \quad \text{(7)}
\]

Towards deriving an easy-to-use form of ELBO, we first notice \( q(y_U) \) can be factorised as in \( q(y_U) = \prod_{u \in U} q(y_u) \) because the neural EA model infers \( y_u \) for \( u \in U \) independently. In addition, we use pseudolikelihood [2] to approximate the joint probability \( p(y_U, y_L) \) for simpler computation, as in Eq. 8. Here, the computation of conditional distribution \( p(y_u | y_{MB_u}) \) can be simplified as in Eq. 9.\(^2\) where \( MB_u = \bigcup_{E \in F_i} E \) is called the Markov Blanket of \( u \) and actually contains \( u \) and its two-hop neighbours.

\[
p(y_U, y_L) = p(y_L) \cdot \prod_{i = 1}^{U} p(y_{U_i}, y_{U_{i-1}}) \approx p(y_L) \cdot \prod_{u \in U} p(y_u | y_{MB_u}) \quad \text{(8)}
\]

Then, we can derive an approximation of ELBO shown in Eq. 10 by introducing the factorised \( q(y_U) \), Eq. 8, and Eq. 9 to Eq. 7.

\[
Q = \log p(y_L) + \sum_{u \in U} \mathbb{E}_{q(y_u)} \left[ \log p(y_u | y_{MB_u}) \right] - \log q(y_U) \quad \text{(10)}
\]

Now, our goal becomes to get optimal \( q(y_u) \) which maximises \( Q \). We solve this problem with coordinate ascent [26]. In particular, we update \( q(y_u) \) for each \( u \in U \) in turn iteratively. Eventually, we only update a single (or a block of) \( q(y_u) \) with Eq. 11, which can be derived from \( \frac{dQ}{dq(y_u)} = 0 \), while keeping the other \( q(y_{e \in U \setminus u}) \) fixed. This process ends until \( q(y_{e \in U}) \) converges.

\[
q^*(y_u) \propto \exp \left( \mathbb{E}_{q(y_{MB_u})} \log p(y_u | y_{MB_u}) \right) \quad \text{(11)}
\]

(\(^2\)The derivation process can be done by introducing Eq. 5 and reducing the items \( g(F_i) \) irrelevant to \( y_e \).)

4.4 Generating Pseudo Mappings

As mentioned, we can derive \( \{q^*(y_u) | u \in U \} \) and \( \{q^*(y_u') | u' \in U \} \) from both KGs. Based on them, we explore three types of strategies for generating pseudo mappings.
We take a few methods to simplify the computation. (1) In Eq. 11, we use $G$ as the source KG to derive dependency-aware EA predictions. (2) Both Eq. 10 and Eq. 11 involve sampling from $q(y_u|y_{MB_u})$ 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.

The whole process of STEA and associated equations of each step are described 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.

| Method     | zh_en Hit@1 | MRR | ja_en Hit@1 | MRR | fr_en Hit@1 | MRR |
|------------|-------------|-----|-------------|-----|-------------|-----|
| IPTransE   | 0.406       | 0.516 | 0.367       | 0.474 | 0.333       | 0.451 |
| GCN-Align  | 0.413       | 0.549 | 0.399       | 0.546 | 0.373       | 0.532 |
| MuGNN      | 0.494       | 0.611 | 0.501       | 0.621 | 0.495       | 0.621 |
| RSN        | 0.508       | 0.591 | 0.507       | 0.590 | 0.516       | 0.605 |
| AliNet     | 0.539       | 0.628 | 0.549       | 0.645 | 0.552       | 0.657 |
| MRAEA      | 0.638       | 0.736 | 0.646       | 0.735 | 0.666       | 0.765 |
| PSR        | 0.702       | 0.781 | 0.698       | 0.782 | 0.731       | 0.807 |
| RREA       | 0.715       | 0.794 | 0.713       | 0.793 | 0.739       | 0.816 |
| Dual-AMN   | 0.731       | 0.799 | 0.726       | 0.799 | 0.756       | 0.827 |

Three datasets are from DBP15K [20], which are cross-lingual and built from different language versions of DBpedia: French-English (fr_en), Chinese-English (zh_en), and Japanese-English (ja_en). Within each dataset, each KG contains around 20K entities, among which 15K are pre-aligned. The other two datasets are from Dwy100K [21], each of which contains two mono-lingual KGs extracted from different sources: dbp_yg extracted from DBpedia and Yago, and dbp_wd extracted from DBpedia and Wikidata. Within each dataset, 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 [13, 31].

Most existing EA works use 30% of the pre-aligned mappings as training data, which however was pointed out unrealistic in practice [30]. In this work, to thoroughly evaluate the self-training methods, we create a few variants of each dataset with different amounts of labelled data – 1%, 5%, 10%, 20%, and 30% of pre-aligned mappings, which are sampled randomly. Within each dataset, all the remaining mappings except the labelled data form the test dataset.

5.2 Metrics

The EA methods typically output a ranked list of candidate counterparts for each entity. Therefore, we choose metrics for measuring the quality of ranking. Following most existing works, we use Hit@k ($k \in \{1, 10\}$) and Mean Reciprocal Rank (MRR) as metrics. Hit@k is the proportion of entities whose ground-truth counterparts rank in top $k$ positions. MRR assigns each prediction result the score $1/\text{Rank}(\text{ground-truth counterpart})$ and then averages them. Higher Hit@k or MRR indicates better performance. Statistical significance is performed using paired two-tailed t-test.

5.3 Baselines

We select the following three self-training methods as baselines: SimTh. It is a general strategy to select the most confident predictions as pseudo-labelled data. Zhu et al. [32] followed this idea and implemented it with weighted loss, which makes the training method coupled with the EA model. Instead, we implement it in a typical way – select the predicted mappings with similarities over a certain threshold and add them to the training data. Obviously, no EA-specific measure is taken to improve the quality of pseudo mappings in this strategy.

OneToOne. The work BootEA [21] proposed to improve the quality of pseudo-mappings by introducing a one-to-one constraint. In each
Align. These neural models vary in performance, KG encoders, We apply each self-training method to different EA models to evaluate its effectiveness and generality. As for our choice of EA models, we select Dual-AMN [15], RREA [17], AliNet [22], and GCN-Align [25]. These neural models vary in performance, KG encoders, and etc. Among them, Dual-AMN and RREA are SOTA models. See Table 1 for a performance summary of existing EA models.

| Anno | Method | Hit@1 | Hit@10 | MRR |
|------|--------|-------|--------|-----|
| Supervised | 0.727 | 0.965 | 0.736 |
| OneToOne | 0.701 | 0.963 | 0.706 |
| MutNearest | 0.678 | 0.963 | 0.706 |

Table 2: Overall performance of STEA and the baselines when running with Dual-AMN. Self-training methods can always achieve much better effectiveness than supervised training; Our method STEA outperforms the baselines significantly; All differences between STEA and other baselines are statistically significant (p < 0.01) except a few cells marked with *.

6 RESULTS AND ANALYSIS

6.1 Comparing STEA with the Baselines

Overall performance. To compare STEA with the baselines, we run these different self-training methods with the SOTA EA model Dual-AMN [15] on five datasets using different annotation amounts ranging from 1% to 30%. Table 2 reports their results measured with three metrics. Our observations include: (1) By comparing the supervised training with each self-training method, we can see that all the self-training methods can bring very significant improvement to the final EA effectiveness under all the settings. Thus, self-training is a better way of training EA model. (2) OneToOne shows an obvious advantage over SimThr, which reveals that one-to-one constraint is useful for deriving pseudo mappings of higher quality. (3) MutNearest outperforms SimThr consistently, while both MutNearest and OneToOne have their own merits in different experimental settings. Note that MutNearest has the advantage of being easier to use in practice. (4) Comparing STEA with each baseline, we can observe that STEA has an obvious advantage in performance under all the settings. The success of STEA verifies it is promising to exploit the dependencies between entities, which is a characteristic of graph data, in devising the self-training framework for EA.
To conclude, self-training can always provide much more effective training for the SOTA EA model than supervised training. Our STEA framework outperforms the existing self-training methods consistently across different datasets and annotation settings.

Generality across different EA models. To check whether these self-training methods work well for different EA models (e.g. in terms of model architecture and performance), we run them with another three EA models: RREA [17], AliNet [22], and GCN-Align [25]. Fig. 4 reports their results on the zh_en dataset (results on the other datasets show a quite similar trend). We have the following findings: (1) The choice of EA model can affect the impact of self-training methods. For example, the existing three self-training methods have slighter impact on AliNet than on the other EA models. (2) Our STEA framework can always outperform the existing self-training methods when running with the different EA models under different datasets and annotation settings. (3) In few annotation settings (e.g. 1% and 5%), the advantage of STEA over the baselines is more obvious than that in rich annotation settings. This is because the baselines depend on the model’s confidence and suffer more from the introduced noise when the EA model is poorly trained. (4) In terms of the final EA effectiveness, the SOTA EA models (i.e. Dual-AMN and RREA) in supervised mode can achieve the best performance in self-training mode as well. In short, the superiority of STEA over the baselines is generic across different EA models.

To check the performance change of self-training methods w.r.t. annotation amounts. The STEA can achieve decent effectiveness in extremely few annotation scenario, while sharp increasement in annotation cannot bring quick improvement.

Figure 4: The performance of self-training methods running with another three EA models: RREA, AliNet, and GCN-Align. The superiority of STEA is genetic across different EA models.

Figure 5: Performance change of self-training methods w.r.t. annotation amounts. The STEA can achieve decent effectiveness in extremely few annotation scenario, while sharp increasement in annotation cannot bring quick improvement.

Figure 6: Precision and recall of pseudo mappings w.r.t. annotation amount. STEA has obvious advantage in both precision and recall regardless of the annotation amount.

In conclusion, self-training remains to be investigated.

Figure 7: Precision and recall of generated pseudo mappings in each iteration when the annotation amount is 1%. STEA is able to suppress the noise without sacrificing recall from the beginning iteration.

supervised training, their performances are still poor when the training data is few; (2) On the contrary, STEA can achieve pretty decent effectiveness. With only 1% labelled data, STEA can achieve comparable performance as supervised learning using 30% labelled data. Similar observations on the other datasets can be obtained from Table 2. From the perspective of self-training for EA, this finding is inspiring. The value of self-training for EA is much beyond what has been realised. Thus, this direction deserves more exploration. (3) The performance of STEA is relatively stable, like reaching a ceiling, even the annotation amount is increased sharply from 1% to 30%. We reckon this phenomenon indicates that random data selection for annotation is not suitable for the self-training based EA methods. Active Learning is a promising direction of smart data selection. Though it has been explored by a few recent works for EA task [11], the combination of active learning with self-training remains to be investigated.

In conclusion, STEA can greatly alleviate the reliance of EA models on annotation. Facing the performance ceiling, active learning for self-training is a promising direction to be studied.

Precision and recall of pseudo mappings. High precision means there is less noise, while high recall of ground-truth mappings means more true training data are added. To get more insight into the
advantage of STEA, we check these two metrics for different self-training methods. Fig. 6 shows these metrics of finally generated pseudo mappings by different self-training methods w.r.t. different annotation amounts on zh_en. We can observe that STEA has a big advantage in both precision and recall in few annotation settings thanks to its dependency-aware predictions. Furthermore, we take a closer look at the generated pseudo mappings in each iteration when the annotation amount is 1%. As shown in Fig. 7, STEA has high precision from the beginning when the EA model has poor effectiveness, while its recall increases gradually from a low level. On the contrary, the baselines can introduce much noise in the beginning and then can hardly get improved in the later iterations. Thus, we reckon the success of STEA comes from: the exploitation of dependency can help suppress the noise without sacrificing high recall of ground-truth mappings.

6.2 Detailed Analysis of the STEA

Comparing strategies of selecting pseudo mappings within STEA. Based on the dependency-aware probabilities, we explore three types of strategies to generate the pseudo mappings – UniThr, BiThr, MutHighestProb. Among them, UniThr has two variants which do compatibility checking only from the first KG (denoted as UniThr (KG1)) or only from the second KG (denoted as UniThr (KG2)). In addition, we use Baseline to denote the best baseline within each annotation setting. Fig. 8 shows their performances on the zh_en dataset. As can be seen: (1) The choice of KG for UniThr may have a big effect on the final performance. (2) BiThr always works better than UniThr (both variants). Meanwhile, it does not need to choose source KG and thus only need to adjust one hyperparameter – threshold. (3) MutHighestProb can achieve the best performance. In addition, it is the easiest strategy to use in practice since it has no hyperparameter. (4) In most cases, these strategies outperform the best baseline.

Necessity of the normalisation module. To normalise the similarities into probabilities, we use one learnable normalisation module based on softmax instead of a simple one, e.g. MinMax scaler. To verify the necessity, we replace our normalisation module with a MinMax scaler in STEA. Table 3 shows the results on dataset zh_en. Obviously, MinMax scaler performs poorly. We think the reason is: the normalised similarity vector by MinMax scaler only contains very small values and the highest value is not distinguishable from the others. Thus, it is not suitable to be used as probability distribution.

Sensitivity to hyperparameter. STEA only has one hyperparameter $K$, which means, for each source entity, only the top $K$ counterpart candidates suggested by the neural EA model will be estimated for compatibility to simplify the computation. In Fig. 9, we show the performance of STEA w.r.t. different $K$ values under different annotation settings. (1) STEA is only sensitive to very small values of $K$ (<5%). (2) Fewer annotations make STEA more sensitive to $K$ with small values (<5%). (3) We prefer to set $K$ as a small value out of the sensitive interval like 10, considering larger $K$ will lead to higher computation cost.

7 CONCLUSION

Entity Alignment is a primary step of fusing different KGs. Though neural EA models have achieved promising performance, their reliance on labelled data is still an open problem. To address this issue, a few self-training strategies have been explored and shown effective in boosting the training of EA models. However, self-training for EA is never studied systematically. In addition, the self-training strategies used in existing works have limited impact because of the introduced noise. In this work, we expand the knowledge about self-training for EA by benchmarking the existing self-training strategies and evaluate them in comparable experimental settings. Furthermore, towards more effective self-training strategy for EA, we propose a new self-training framework named STEA. This framework features in exploiting the dependencies between entities to detect suspicious mappings and improve EA predictions. Based on the derived dependency-aware EA predictions, we further explored different ways for generating pseudo mappings to be included in self-training. We empirically show that STEA outperforms the existing methods with a big margin across different datasets, annotation amounts, and neural EA models. STEA can greatly reduce the reliance of EA on annotations. In particular, STEA using 1% labelled data can achieve decent effectiveness, which is equivalent to supervised training using 30% labelled data.

In future, we plan to investigate the combination of active learning and self-training for EA. As shown, in the self-training based EA, random data selection is very inefficient in improving the EA performance. With active learning, we aim to reach the desired EA performance with the least annotation effort.

ACKNOWLEDGMENTS

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).
