Matching Knowledge Graphs in Entity Embedding Spaces: An Experimental Study

Weixin Zeng, Xiang Zhao, Zhen Tan, Jiuyang Tang, Xueqi Cheng

Abstract—Entity alignment (EA) identifies equivalent entities that locate in different knowledge graphs (KGs), and has attracted growing research interests over the last few years with the advancement of KG embedding techniques. Although a pile of embedding-based EA frameworks have been developed, they mainly focus on improving the performance of *entity representation learning*, while largely overlook the subsequent stage that *matches KGs in entity embedding spaces*. Nevertheless, accurately matching entities based on learned entity representations is crucial to the overall alignment performance, as it coordinates individual alignment decisions and determines the global matching result. Hence, it is essential to understand how well existing solutions for matching KGs in entity embedding spaces perform on present benchmarks, as well as their strengths and weaknesses. To this end, in this article we provide a comprehensive survey and evaluation of matching algorithms for KGs in entity embedding spaces in terms of effectiveness and efficiency on both classic settings and new scenarios that better mirror real-life challenges. Based on in-depth analysis, we provide useful insights into the design trade-offs and good paradigms of existing works, and suggest promising directions for future development.

1 Introduction

Matching data instances that refer to the same real-world entity is a long-standing problem. It establishes the connections among multiple data sources, and is critical to data integration and cleaning [39]. Therefore, the task has been actively studied; for instance, in the database community, various entity matching (EM) (and entity resolution (ER)) strategies are proposed to train a (supervised) classifier to predict whether a pair of data records match [10], [39].

Recently, due to the emergence and proliferation of knowledge graphs (KGs), matching entities in KGs draws much attention from both academia and industries. Distinct from traditional data matching, it brings its own challenges. Particularly, it underlines the use of KGs' structures for matching, and manifests unique characteristics of data, e.g., imbalanced class distribution, few attributive textual information, etc. In consequence, although viable, following traditional EM pipeline, it is hard to train an effective classifier that can infer the equivalence between entities. Thus, much effort has been dedicated to specifically addressing the matching of entities in KGs, which is also referred to as *entity alignment* (EA).

Nevertheless, early solutions to EA are mainly unsupervised [25], [48], i.e., no labeled data is assumed. They utilize discriminative features of entities (e.g., entity descriptions and relational structures) to infer the equivalent entity pair, which are, however, embarrassed by the heterogeneity of independently-constructed KGs [50].

To mitigate this issue, recent solutions to EA employ a few labeled pairs as seeds to guide the learning and prediction [9], [16], [31], [43], [54]. In short, they embed the symbolic representations of KGs as low-dimensional vectors in a way such that the semantic relatedness of entities is captured by the geometrical structures of embedding spaces [4], where the seed pairs are leveraged to produce unified entity representations. In the testing stage, they match entities based on the unified entity embeddings. They are coined as *embedding-based* EA methods, which have exhibited state-of-the-art performance on existing benchmarks.

To be more specific, the embedding-based EA [1] pipeline can be roughly divided into two major stages, i.e., *representation learning* and *matching KGs in entity embedding spaces* (or *embedding matching* for short). While the former encodes the KG structures into low-dimensional vectors and establishes connections between independent KGs via the calibration or transformation of (seed) *entity embeddings* [50], the latter computes pairwise scores between source and target entities based on such embeddings and then makes alignment decisions according to the pairwise scores. Although this field has been actively explored, existing efforts are mainly devoted to the *representation learning* stage [19], [30], [70], while *embedding matching* has not raised many attentions until very recently [35], [62]. The majority of existing EA solutions adopt a simple algorithm to realize this stage, i.e., DInf, which first leverages common similarity metrics such as cosine similarity to calculate the pairwise similarity scores between entity embeddings, and then matches a source entity to its most similar target entity according to the pairwise scores [54]. Nevertheless, it is evident that such an intuitive strategy can merely reach local optimums for individual entities and completely overlooks the (global) interdependence among the matching decisions for different entities [64].

To address the shortcomings of DInf, advanced strategies are devised [13], [50], [57], [62], [64], [65]. While some of them inject the modeling of global interdependence into the computation of pairwise scores [13], [50], [62], some directly improve the alignment decision-making process by imposing collective matching constraints [57], [64], [65]. These efforts demonstrate the significance of *matching KGs in entity embedding spaces* from at least three major aspects: (1) It is an indispensable step of EA, which takes as input the entity embeddings (generated by

- *Weixin Zeng, Xiang Zhao, Zhen Tan, Jiuyang Tang are with Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, China. E-mail:{zengweixin13, xiangzhao, tanzhen08a, jiuyang_tang}@nudt.edu.cn.*
- *Xueqi Cheng is with Institute of Computing Technology, CAS, China. E-mail: cxq@ict.ac.cn.*

1. In the rest of the paper, we use *EA* to refer to embedding-based EA solutions, and use *conventional EA* for the early solutions.

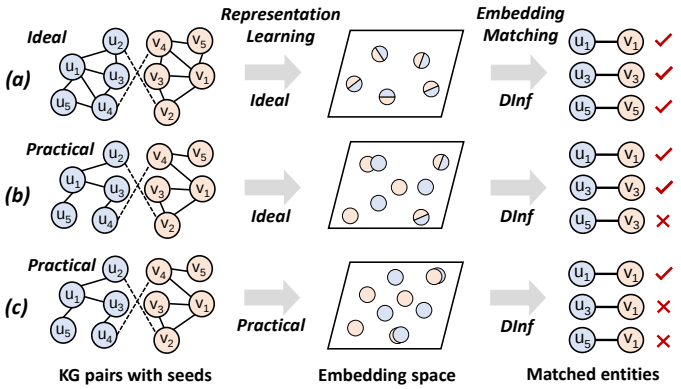


Fig. 1. Three cases of EA. Dashes lines between KGs denote the seed entity pairs. Entities with the same subscripts are equivalent. In the embedding space, the circles with two colors represent that the corresponding entities in the two KGs have the same embeddings.

the representation learning stage), and outputs matched entity pairs; (2) Its performance is crucial to the overall EA results, e.g., an effective algorithm can improve the alignment results by up to 88% [62]; and (3) It empowers EA with explainability, as it unveils the decision-making process of alignment. We use Example 1 to further illustrate the significance of the embedding matching process.

Example 1. *Figure 1 presents three representative cases of EA. The KG pairs to be aligned are first encoded into embeddings via the representation learning models. Next, the embedding matching algorithms produce the matched entity pairs based on the embeddings. In the most ideal case where two KGs are identical, e.g., case (a), with an ideal representation learning model, equivalent entities would be embedded into exactly the same place in the low-dimensional space, and using the simple* DInf *algorithm would attain perfect results. Nevertheless, in the majority of practical scenarios, e.g., case (b) and (c), the two KGs have high structure heterogeneity. As thus, even an ideal representation learning model might generate different embeddings for equivalent entities. In this case, adopting the simple* DInf *strategy is likely to produce false entity pairs, such as* (u_5, v_3) *in case (b).*

Worse still, as pointed out in previous works [50], [68], existing representation learning methods for EA cannot fully capture the structural information (possibly due to their inner design mechanisms, or their incapability of dealing with scarce supervision signals). Under these settings, e.g., case (c), the distribution of entity embeddings in the low-dimensional space would become irregular, where the simple embedding matching algorithm DInf *would fall short, i.e., producing incorrect entity pairs* (u_3, v_1) *and* (u_5, v_1)*. As thus, in these practical cases, an effective embedding matching algorithm is crucial to inferring the correct matches. For instance, by exploiting the collective embedding matching algorithm that imposes the 1-to-1 alignment constraint, the correct matches, i.e.,* (u_3, v_3) *and* (u_5, v_5)*, are likely to be restored.*

While the study on matching KGs in entity embedding spaces is rapidly progressing, there is no systematic survey or comparison of these solutions [50]. We do notice that there are several survey papers covering embedding-based EA frameworks [50], [61], [66], [67], [68], whereas they all briefly introduce the embedding matching module (mostly only mentioning the DInf algorithm). In this article, we aim to fill in this gap by surveying current solutions for matching KGs in entity embedding spaces and providing a comprehensive evaluation of these methods with the following features:

(1) Systematic survey and fair comparison. Albeit essential to the alignment performance, existing embedding matching strategies have yet not been compared directly. Instead, they are integrated with representation learning models, and then evaluated and compared with each other (as a whole). This, however, cannot provide a fair comparison of the embedding matching strategies themselves, since the difference among them can be offset by other influential factors, such as the choices of representation learning models or input features. Therefore, in this work, we exclude irrelevant factors and provide a fair comparison of current matching algorithms for KGs in entity embedding spaces at both theoretical and empirical levels.

(2) Comprehensive evaluation and detailed discussion. To fully appreciate the effectiveness of embedding matching strategies, we conduct extensive experiments on a wide range of EA settings, i.e., with different representation learning models, with various input features, and on datasets at different scales. We also analyze the complexity of these algorithms and evaluate their efficiency/scalability under each experimental setting. Based on the empirical results, we discuss to reveal strengths and weaknesses.

(3) New experimental settings and insights. Through empirical evaluation and analysis, we discover that the current mainstream evaluation setting, i.e., 1-to-1 constrained EA, oversimplifies the real-life alignment scenarios. As thus, we identify two experimental settings that better reflect the challenges in practice, i.e., alignment with unmatchable entities, as well as a new setting of *non 1-to-1 alignment*. We compare the embedding matching algorithms under these challenging settings to provide further insights.

Contributions. We make the following contributions:

- We systematically and comprehensively survey and compare state-of-the-art algorithms for *matching KGs in entity embedding spaces* (Section 3).
- We evaluate and compare the state-of-the-art embedding matching algorithms on a wide range of EA datasets and settings, as well as reveal their strengths and weaknesses. The codes of these algorithms are organized and integrated into an open-source library, `EntMatcher`, publicly available at https://github.com/DexterZeng/EntMatcher (Section 4).
- We identify experiment settings that better mirror real-life challenges and construct a new benchmark dataset, where deeper insights into the algorithms are obtained via empirical evaluations (Section 5).
- Based on our evaluation and analysis, we provide useful insights into the design trade-offs of existing works, and suggest promising directions for the future development of matching KGs in entity embedding spaces (Section 6).

2 Preliminaries

In this section, we first present the task formulation of EA and its general framework. Next, we introduce the studies related

to the topic of this article—*matching KGs in entity embedding spaces*, and clarify the scope of this study. Finally, we present the key assumptions of embedding-based EA.

2.1 Task Formulation and Framework

Task formulation. A KG $\mathcal{G}$ is composed of triples $\{(s, p, o)\}$, where $s, o \in \mathcal{E}$ represent entities, $p \in \mathcal{P}$ denotes the predicate (relation). Given a source KG $\mathcal{G}_s$, a target KG $\mathcal{G}_t$, the task of EA is formulated as discovering new (equivalent) entity pairs $\mathcal{M} = \{(u, v)|u \in \mathcal{E}_s, v \in \mathcal{E}_t, u \Leftrightarrow v\}$ by using pre-annotated (seed) entity pairs $\mathcal{S}$ as anchors, where $\Leftrightarrow$ represents the equivalence between entities, $\mathcal{E}_s$ and $\mathcal{E}_t$ denote the entity sets in $\mathcal{G}_s$ and $\mathcal{G}_t$, respectively.

General framework. The pipeline of state-of-the-art embedding-based EA solutions can be divided into two stages, i.e., *representation learning* and *embedding matching*, as shown in Figure 2. The general algorithm can be found in Algorithm 1.

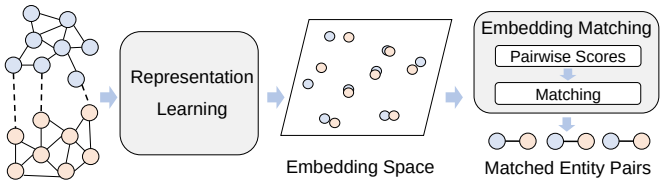


Fig. 2. The pipeline of embedding-based EA. Dashed lines denote the pre-annotated alignment links.

Algorithm 1: General Algorithm of Embedding-based EA.

Input : Source and target KGs: $\mathcal{G}_s$, $\mathcal{G}_t$; Seed pairs: $\mathcal{S}$
Output : Aligned entity pairs: $\mathcal{M}$

$\boldsymbol{E} \leftarrow$ Representation_Learning($\mathcal{G}_s$, $\mathcal{G}_t$, $\mathcal{S}$);
$\mathcal{M} \leftarrow$ Embedding_Matching($\mathcal{E}_s$, $\mathcal{E}_t$, $\boldsymbol{E}$);
return $\mathcal{M}$;

The majority of studies on EA are devoted to the *representation learning* stage. They first utilize KG embedding techniques such as TransE [4] and GCN [23] to capture the KG structure information and generate entity structural representations. Next, based on the assumption that equivalent entities from different KGs possess similar neighboring KG structures (and in turn similar embeddings), they leverage the seed entity pairs as anchors and progressively project individual KG embeddings into a unified space through training, resulting in the unified entity representations $\boldsymbol{E}$ [2]. There have already been several survey papers concentrating on representation learning approaches for EA, and we refer the interested readers to these works [2], [50], [66], [68].

Next, we introduce the *embedding matching* process—the focus of this article, as well as its related works.

2.2 Related Work and Scope

Matching KGs in entity embedding spaces. After obtaining the unified entity representations $\boldsymbol{E}$ where equivalent entities from different KGs are assumed to have similar embeddings, the *embedding matching* stage (also frequently referred to as *alignment inference* stage [50]) produces alignment results by comparing the embeddings of entities from different KGs. Concretely, it first calculates the pairwise scores between source and target entity embeddings according to a specific metric [3]. The pairwise scores are then organized into matrix form as $\boldsymbol{S}$. Next, according to the pairwise scores, various matching algorithms are put forward to align entities. The most common algorithm is Greedy, described in Algorithm 2. It directly matches a source entity to the target entity that possesses the highest pairwise score according to $\boldsymbol{S}$. Over the last few years, advanced solutions [13], [17], [34], [35], [40], [50], [57], [60], [62], [64], [65], [69] are devised to improve the embedding matching performance, and in this work, we focus on surveying and comparing these algorithms for matching KGs in entity embedding spaces.

Algorithm 2: Greedy $(\mathcal{E}_s, \mathcal{E}_t, \boldsymbol{S})$

Input : Source and target entity sets: $\mathcal{E}_s$, $\mathcal{E}_t$; The similarity matrix of pairwise scores: $\boldsymbol{S}$
Output : Matched entity pairs: $\mathcal{M}$

for $u \in \mathcal{E}_s$ **do**
  $v^* = \arg\max_{v \in \mathcal{E}_t} \boldsymbol{S}(u, v)$;
  $\mathcal{M} \leftarrow \mathcal{M} \cup \{(u, v^*)\}$;
return $\mathcal{M}$;

Matching KGs in symbolic spaces. Before the emergence of embedding-based EA, there have already been many conventional frameworks that match KGs in symbolic spaces [20], [47], [48]. While some are based on equivalence reasoning mandated by OWL semantics [20], some leverage similarity computation to compare the symbolic features of entities [48]. However, these solutions are not comparable to algorithms for matching KGs in entity embedding spaces, as (1) they cover *both* the representation learning and embedding matching stages in embedding-based EA; and (2) the inputs are different from those of embedding matching algorithms. Thus, we do not include them in our experimental evaluation, while they have already been compared in the survey papers covering the overall embedding-based EA frameworks [50], [68].

The matching of relations (or ontology) between KGs has also been studied by prior symbolic works [47], [48]. Nevertheless, compared with entities, they are usually in smaller amounts, of various granularities [42], and under-explored in embedding-based approaches [59]. Hence, in this work, we exclude relevant studies on this topic and focus on the matching of *entities*.

The task of entity resolution (ER) [10], [18], [41], also known as entity matching, deduplication or record linkage, can be regarded as the general case of EA [68]. It assumes that the input is relational data, and each data object usually has a large amount of textual information described in multiple attributes. Nevertheless, in this article, we focus on EA approaches, which strive to align *KGs* and mainly rely on *graph representation*

2. Indeed there are a few exceptions, which instead learn a mapping function between individual embedding spaces [50]. However, the subsequent steps still require mapping between spaces and operate on a "unified" one, e.g., target entity embeddings.

3. Under certain metrics such as cosine similarity (resp., Euclidean distance), the larger (resp., smaller) the pairwise scores, the higher the probability that two entities are equivalent. In this work, w.l.o.g., we adopt the former expression and consider that higher pairwise scores are preferred.

learning techniques to model the KG structure and generate entity structural *embeddings* for alignment. Therefore, the discussion and comparison with ER solutions is beyond the scope of this work.

Matching data instances via deep learning. Entity matching (EM) between databases have also been greatly advanced by utilizing pre-trained language models for expressive contextualization of database records [11], [39]. These deep learning (DL) based EM solutions devise end-to-end neural models to learn to classify an entity pair into matching or non-matching, and then feed the test entity pairs into the trained models to obtain classification results [5], [29], [39]. Nevertheless, this procedure is different from the focus of our study, as both of its training and testing stage involve representation learning and matching. Besides, these solutions are not suitable for matching KGs in entity embedding space, since (1) they require adequate labeled data to train the neural classification models, but the training data in EA is much less than the testing ones, which could result in the overfitting issue; (2) they would suffer from severe class imbalance in EA, where an entity and all of its nonequivalent entities in another KG would constitute many negative samples, while there is usually one positive sample for this entity; (3) they depend on the attributive text information between data records for training, while EA underlines the use of KG structure, which could provide much less useful features for model training. In the experiment, we adapt DL-based EM models to tackle EA, and the results are not promising. This will be further discussed in Section 4.3.

TABLE 1
Comparison with existing surveys on EA. The focus of each work is denoted with ✓.

Surveys	Repre. Learn.	Embed. Match.	Papers Included
Zhao et al. [68]	✓	✗	Before 2020
Sun et al. [50]	✓	✗	Before 2020
Zhang et al. [67]	✓	✗	Before 2020
Zeng et al. [61]	✓	✗	Before 2021
Zhang et al. [66]	✓	✗	Before 2021
Our work	✗	✓	Before 2023

Existing surveys on EA. There are several survey papers covering EA frameworks [50], [61], [66], [67], [68], which are summarized in Table 1. Some articles provide high-level discussion of embedding-based EA frameworks, experimentally evaluate and compare these works, and offer guidelines for potential practitioners [50], [67], [68]. Specifically, Zhao et al. propose a general EA framework to encompass existing works, and then evaluate them under a wide range of settings. Nevertheless, they only briefly mention DInf and SMat in the embedding matching stage [68]. Sun et al. survey EA approaches and develop an open-source library to evaluate existing works. However, they merely introduce DInf, SMat and CSLS, and overlook the comparison among these algorithms. Besides, they point out that current approaches put in their main efforts in learning expressive embeddings to capture entity features while ignore the alignment inference (i.e., embedding matching) stage [50]. Zhang et al. empirically evaluate state-of-the-art embedding-based EA methods in an industrial context, and particularly investigate the influence of the sizes and biases in seed mappings. They evaluate each method as a whole and do not mention the embedding matching process [67].

Two recent survey papers include the latest efforts on embedding-based EA and give more self-contained explanation on each technique. Zhang et al. provide a tutorial-type survey, while for embedding matching, they merely introduce the nearest neighbor search strategy, i.e., DInf [66]. Zeng et al. mainly introduce representation learning methods and their applications on EA, while neglect the embedding matching stage [61].

In all, existing EA survey articles focus on the **representation learning** process and briefly introduce the embedding matching module (mostly only mentioning the DInf algorithm), while in this work we systematically survey and empirically evaluate the algorithms designed for the **embedding matching** process in KG alignment, and present comprehensive results and insightful discussions.

Scope of this work. This study aims to survey and empirically compare the algorithms for matching KGs in *entity embedding* spaces, i.e., various implementations of Embedding_Matching( ) in Algorithm 1, on a wide range of EA experimental settings.

2.3 Key Assumptions

Notably, existing embedding-based EA solutions have a **fundamental assumption**; that is, the equivalent entities in different KGs possess similar (ideally, isomorphic) neighboring structures. Under such an assumption, effective representation learning models would transform the structures of equivalent entities into similar entity embeddings. As thus, based on the entity embeddings, the embedding matching stage would assign higher (resp., lower) pairwise similarity scores to the equivalent (resp., nonequivalent) entity pairs, and finally make accurate alignment decisions via the coordination according to pairwise scores.

Besides, current EA evaluation settings assume that the entities in different KGs conform to the 1-to-1 constraint. That is, each $u \in \mathcal{E}_s$ has *one and only one* equivalent entity $v \in \mathcal{E}_t$, and vice versa. However, we contend that this assumption is in fact impractical and provide detailed experiments and discussions in Section 5.2.

3 Algorithms for Matching KGs in Entity Embedding Spaces

In this section, we introduce the algorithms for matching KGs in entity embedding spaces, i.e., Embedding_Matching( ) in Algorithm 1.

3.1 Overview

We first provide the overview and comparison of matching algorithms for KGs in entity embedding spaces in Table 2. As mentioned in Section 2, embedding matching comprises two stages—**pairwise score** computation and **matching**. The baseline approach DInf adopts existing similarity metrics to calculate the similarity between entity embeddings and generate the pairwise scores in the first stage, and then it leverages Greedy for matching. In pursuit of better alignment performance, more advanced embedding matching strategies

TABLE 2
Overview and comparison of state-of-the-art algorithms for matching KGs in entity embedding spaces. Note that we estimate the order of magnitude of the time and space complexity.

Models	Pairwise Scores	Matching	1-to-1	Direction	Time Comp.	Space Comp.
DInf [6], [32], [70]	Similarity metric	Greedy	✗	Unidirectional	$O(n^2)$	$O(n^2)$
CSLS [15], [34], [50], [51]	CSLS	Greedy	✗	Partially bidirectional	$O(n^2)$	$O(n^2)$
RInf [62]	Preference modeling	Greedy	✗	Partially bidirectional	$O(n^2 \lg n)$	$O(n^2)$
Sink. [13], [15], [17]	Sinkhorn operation	Greedy	Partially	Partially bidirectional	$O(ln^2)$	$O(n^2)$
Hun. [35], [57]	Similarity metric	Hungarian	✓	Bidirectional	$O(n^3)$	$O(n^2)$
SMat [64], [69]	Similarity metric	Gale-Shapley	✓	Bidirectional	$O(n^2 \lg n)$	$O(n^2)$
RL [65]	Similarity metric	Reinforcement learning	Partially	Unidirectional	/	$O(n^2)$

are put forward. While some (i.e., CSLS, RInf and Sink.) optimize the **pairwise score** computation process and produce more *accurate* pairwise scores, some (i.e., Hun., SMat and RL) take into account the global alignment dynamics, rather than greedily pursue the local optimum for each entity, during the **matching** process, where more correct matches could be generated according to the coordination under the global constraint.

We further identify two notable characteristics of matching KGs in entity embedding spaces, i.e., whether the matching leverages the **1-to-1** constraint, and the **direction** of the matching. Regarding the former, Hun. and SMat explicitly exert the 1-to-1 constraint on the matching process. RL relaxes the strict 1-to-1 constraint by allowing non 1-to-1 matches. The greedy strategies, however, normally do not take into consideration this constraint, except for Sink., which implicitly implements the 1-to-1 constraint in a progressive manner when calculating the pairwise scores. As for the **direction** of matching, Greedy only considers a single direction at a time and overlooks the influence from the reverse direction. As thus, the resultant source-to-target alignment results are not necessarily equal to the target-to-source ones. By improving the pairwise score computation, CSLS, RInf and Sink. are actually modeling and integrating the bidirectional alignments, whereas they still adopt Greedy to produce final results. For non-greedy methods, Hun. and SMat fully consider the bidirectional alignments and produce a matching agreed by both directions, while RL is unidirectional.

Next, we describe these methods in detail [4].

3.2 Simple Embedding Matching

DInf is the most common implementation of Embedding_Matching( ), described in Algorithm 3. Assume both KGs contain n entities. The time and space complexity of DInf is $O(n^2)$.

Algorithm 3: DInf$(\mathcal{E}_s, \mathcal{E}_t, \boldsymbol{E})$

Input : Source and target entity sets: $\mathcal{E}_s$, $\mathcal{E}_t$; Unified entity embeddings: $\boldsymbol{E}$
Output : Matched entity pairs: $\mathcal{M}$

Derive similarity matrix $\boldsymbol{S}$ based on $\boldsymbol{E}$;
$\mathcal{M} \leftarrow$ Greedy $(\mathcal{E}_s, \mathcal{E}_t, \boldsymbol{S})$;
return $\mathcal{M}$;

4. We omit the algorithmic description of the classical algorithms (e.g., Hungarian [24] and Gale-Shapley [46]) and the neural model (i.e., RL [38]) in the interest of space.

3.3 CSLS Algorithm

The cross-domain similarity local scaling (CSLS) algorithm [26] is introduced to mitigate the hubness and isolation issues of entity embeddings in EA [50]. The hubness issue refers to the phenomenon where some entities (known as hubs) frequently appear as the top-1 most similar entities of other entities in the vector space, while the isolation issue means that there exist some outliers isolated from any point clusters. As thus, CSLS increases the similarity associated with isolated entity embeddings, and conversely decreases the ones of vectors lying in dense areas [26]. Formally, the CSLS pairwise score between source entity u and target entity v is:

$$\text{CSLS}(u, v) = 2\boldsymbol{S}(u, v) - \phi(u) - \phi(v)^\top, \tag{1}$$

where $\boldsymbol{S}$ is the similarity matrix derived from $\boldsymbol{E}$ using similarity metrics, $\phi(u) = \frac{1}{k}\sum_{v' \in \mathcal{N}_u} \boldsymbol{S}(u, v')$ is the mean similarity score between the source entity u and its top-k most similar entities $\mathcal{N}_u$ in the target KG, and $\phi(v)$ is defined similarly. The mean similarity scores of all source and target entities are denoted in vector form as ϕ_s and ϕ_t, respectively. To generate the matched entity pairs, it further applies Greedy on the CSLS matrix (i.e., $\boldsymbol{S}_{\text{CSLS}}$). Algorithm 4 describes the detailed procedure of CSLS. Notably, Li et al. put forward Graph Interactive Divergence (GID) to compute the similarity score, which in essence works in the same way as CSLS according to its code implementation [28].

Algorithm 4: CSLS $(\mathcal{E}_s, \mathcal{E}_t, \boldsymbol{E}, k)$

Input : Source and target entity sets: $\mathcal{E}_s$, $\mathcal{E}_t$; Unified entity embeddings: $\boldsymbol{E}$; Hyper-parameter: k
Output : Matched entity pairs: $\mathcal{M}$

Derive similarity matrix $\boldsymbol{S}$ based on $\boldsymbol{E}$;
Calculate the mean values of top-k similarity scores of entities in $\mathcal{E}_s$ and $\mathcal{E}_t$, resulting in ϕ_s and ϕ_t, respectively;
$\boldsymbol{S}_{\text{CSLS}} = 2\boldsymbol{S} - \phi_s - \phi_t^\top$;
$\mathcal{M} \leftarrow$ Greedy $(\mathcal{E}_s, \mathcal{E}_t, \boldsymbol{S}_{\text{CSLS}})$;
return $\mathcal{M}$;

Complexity. The time and space complexity are $O(n^2)$. Practically, it requires more time and space than DInf, as it needs to generate the additional CSLS matrix.

3.4 Reciprocal Embedding Matching

Zeng et al. [62] formulate EA task as the reciprocal recommendation process [44] and offer a reciprocal embedding

matching strategy RInf to model and integrate the bidirectional preferences of entities when inferring the matching results. Formally, it defines the pairwise score of source entity u towards target entity v as:

$$p_{u,v} = \boldsymbol{S}(u,v) - \max_{u' \in \mathcal{E}_s} \boldsymbol{S}(v,u') + 1, \tag{2}$$

where $\boldsymbol{S}$ is the similarity matrix derived from $\boldsymbol{E}$, $0 \leq p_{u,v} \leq 1$, and a larger $p_{u,v}$ denotes a higher degree of preference. As such, the matrix forms of the source-to-target and target-to-source preference scores are denoted as $\boldsymbol{P}_{s,t}$ and $\boldsymbol{P}_{t,s}$, respectively. Next, it converts the preference matrix $\boldsymbol{P}$ into the ranking matrix $\boldsymbol{R}$, and then averages the two ranking matrices, resulting in the reciprocal preference matrix $\boldsymbol{P}_{s\leftrightarrow t}$ that encodes the bidirectional alignment information. Finally, it adopts Greedy to generate the matched entity pairs.

Algorithm 5: RInf$(\mathcal{E}_s, \mathcal{E}_t, \boldsymbol{E})$

Input : Source and target entity sets: $\mathcal{E}_s$, $\mathcal{E}_t$; Unified entity embeddings: $\boldsymbol{E}$
Output : Matched entity pairs: $\mathcal{M}$

Derive similarity matrix $\boldsymbol{S}$ based on $\boldsymbol{E}$;
for $u \in \mathcal{E}_s$ **do**
  for $v \in \mathcal{E}_t$ **do**
    Calculate $p_{u,v}$ and $p_{v,u}$ (cf. Equation (2));
Collect the preference scores, resulting in $\boldsymbol{P}_{s,t}$ and $\boldsymbol{P}_{t,s}$;
Convert $\boldsymbol{P}_{s,t}$ and $\boldsymbol{P}_{t,s}$ into $\boldsymbol{R}_{s,t}$ and $\boldsymbol{R}_{t,s}$, respectively;
$\boldsymbol{P}_{s\leftrightarrow t} = (\boldsymbol{R}_{s,t} + \boldsymbol{R}_{t,s}^{\top})/2$;
$\mathcal{M} \leftarrow$ Greedy $(\mathcal{E}_s, \mathcal{E}_t, -\boldsymbol{P}_{s\leftrightarrow t})$;
return $\mathcal{M}$;

Complexity. Algorithm 5 describes the detailed procedure of RInf. The time complexity is $O(n^2 \lg n)$ [62]. The space complexity is $O(n^2)$. Practically, it requires more space than DInf and CSLS, due to the computation of similarity, preference, and ranking matrices. Noteworthily, two variant methods, i.e., RInf-wr and RInf-pb, are proposed to reduce the memory and time consumption brought by the reciprocal modeling. More details can be found in [62].

3.5 Embedding Matching as Assignment

Some very recent studies [35], [57] propose to model the embedding matching process as the linear assignment problem. They first use similarity metrics to calculate pairwise similarity scores based on $\boldsymbol{E}$. Then they adopt the Hungarian algorithm [24] to solve the task of assigning source entities to target entities according to the pairwise scores. The objective is to maximize the sum of the pairwise similarity scores of the final matched entity pairs while observing the 1-to-1 assignment constraint. In this work, we use the Hungarian algorithm implemented by Jonker and Volgenant [21] and denote it as Hun. $(\mathcal{E}_s, \mathcal{E}_t, \boldsymbol{E})$.

Besides, the Sinkhorn operation [37] (or Sink. for short) is also adopted to solve the assignment problem [13], [17], [35], which converts the similarity matrix $\boldsymbol{S}$ into a doubly stochastic matrix $\boldsymbol{S}_{sinkhorn}$ that encodes the entity correspondence information. Specifically,

$$\begin{aligned} &Sinkhorn^l(\boldsymbol{S}) = \Gamma_c(\Gamma_r(Sinkhorn^{l-1}(\boldsymbol{S}))); \\ &\boldsymbol{S}_{sinkhorn} = \lim_{l\to\infty} Sinkhorn^l(\boldsymbol{S}), \end{aligned} \tag{3}$$

where $Sinkhorn^0(\boldsymbol{S}) = \exp(\boldsymbol{S})$, Γ_c and Γ_r refer to the column and row-wise normalization operators of a matrix. Since the number of iterations l is limited, the Sinkhorn operation can only obtain an approximate 1-to-1 assignment solution in practice [35]. Then $\boldsymbol{S}_{sinkhorn}$ is forwarded to Greedy to obtain the alignment results.

Algorithm 6: Sink. $(\mathcal{E}_s, \mathcal{E}_t, \boldsymbol{E}, l)$

Input : Source and target entity sets: $\mathcal{E}_s$, $\mathcal{E}_t$; Unified entity embeddings: $\boldsymbol{E}$; Hyper-parameter: l
Output : Matched entity pairs: $\mathcal{M}$

Derive similarity matrix $\boldsymbol{S}$ based on $\boldsymbol{E}$;
$\boldsymbol{S}_{sinkhorn} = Sinkhorn^l(\boldsymbol{S})$ (cf. Equation (3));
$\mathcal{M} \leftarrow$ Greedy $(\mathcal{E}_s, \mathcal{E}_t, \boldsymbol{S}_{sinkhorn})$;
return $\mathcal{M}$;

Complexity. For Hun., the time complexity is $O(n^3)$, and the space complexity is $O(n^2)$. Algorithm 5 describes the procedure of Sink.. The time complexity of Sink. is $O(ln^2)$ [35], and the space complexity is $O(n^2)$. In practice, both algorithms require more space than DInf, since they need to store the intermediate results.

3.6 Stable Embedding Matching

In order to consider the interdependence among alignment decisions, the embedding matching process is formulated as the stable matching problem [14] by [64], [69]. It is proved that for any two sets of members with the same size, each of whom provides a ranking of the members in the opposing set, there exists a bijection of the two sets such that no pair of two members from the opposite side would prefer to be matched to each other rather than their assigned partners [12]. Specifically, these works first produce the similarity matrix $\boldsymbol{S}$ based on $\boldsymbol{E}$ using similarity metrics. Next, they generate the rankings of members in the opposing set according to the pairwise similarity scores. Finally, they use the Gale-Shapley algorithm [46] to solve the stable matching problem. This procedure is denoted as SMat $(\mathcal{E}_s, \mathcal{E}_t, \boldsymbol{E})$.

Complexity. SMat has time complexity of $O(n^2 \lg n)$ (since for each entity, the ranking of entities in the opposite side needs to be computed) and space complexity of $O(n^2)$.

3.7 RL-based Embedding Matching

The embedding matching process is cast to the classic sequence decision problem by [65]. Given a sequence of source entities (and their embeddings), the goal of the sequence decision problem is to decide to which target entity each source entity aligns. It devises a reinforcement learning (RL)–based framework to learn to optimize the decision-making for all entities, rather than optimize every single decision separately. Under the RL-based framework, a new coordination strategy that involves the coherence and exclusiveness constraints is implemented. While *coherence* aims to keep the EA decisions

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2023.3272584

coherent for closely-related entities, *exclusiveness* aims to avoid assigning the same target entity to multiple source entities, which requires that, if an entity is already matched, it is less likely to be matched to other entities. The general procedure is shown in algorithmic form in Appendix A due to the limit of space, and more details can be found in the original paper [65].

Complexity. It is difficult to deduce the time complexity for this neural RL model. Instead, we provide the empirical time costs in experiments. The space complexity is $O(n^2)$.

4 Main Experiments

In this section, we compare the algorithms for matching KGs in entity embedding spaces on the mainstream EA evaluation setting (1-to-1 alignment).

4.1 EntMatcher: An Open-source Library

To ensure comparability, we re-implemented all compared algorithms using Python under a unified framework and established an open-source library, `EntMatcher` [5]. The architecture of `EntMatcher` library is presented in the blue block of Figure 3, which takes as input unified entity embeddings $\boldsymbol{E}$ and produces the matched entity pairs. It has the following three major features:

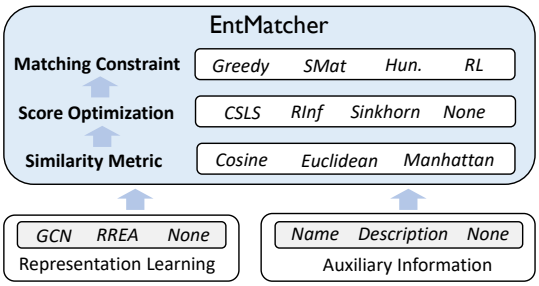


Fig. 3. Architecture of the `EntMatcher` library and additional modules required by the experimental evaluation.

Loosely-coupled design. There are three independent modules in `EntMatcher`, and we have implemented the representative methods in each module. Users are free to combine the techniques in each module to develop new approaches, or to implement their new designs by following the templates in modules.

Reproduction of existing approaches. To support our experimental study, we tried our best to re-implement all existing algorithms by using `EntMatcher`. For instance, the combination of cosine similarity, CSLS, and Greedy reproduces the CSLS algorithm in Section 3.3; and the combination of cosine similarity, None, and Hun. reproduces the Hun. algorithm in Section 3.5. The specific hyper-parameter settings are elaborated in Section 4.2.

Flexible integration with other modules in EA. `EntMatcher` is highly flexible, which can be directly called during the development of standalone EA approaches. Besides, users may also use `EntMatcher` as the backbone and call other modules. For instance, to conduct the experimental evaluations in this work, we implemented the representation learning and auxiliary information modules to generate the unified entity embeddings $\boldsymbol{E}$, as shown in the white blocks of Figure 3. More details are elaborated in the next subsection. Finally, `EntMatcher` is also compatible with existing open-source EA libraries (that mainly focus on representation learning) such as OpenEA [6] and EAkit [7].

4.2 Experimental Settings

Current EA evaluation setting assumes that the entities in source and target KGs are 1-to-1 matched (cf. Section 2.3). Although this assumption simplifies the real-word scenarios where some entities are unmatchable or some might be aligned to multiple entities on the other side, it indeed reflects the core challenge of EA. Therefore, following existing literature, we mainly compare the embedding matching algorithms under this setting, and postpone the evaluation on the challenging real-life scenarios to Section 5.

Datasets. We used popular EA benchmarks for evaluation: **(1)** **`DBP15K`**, which comprises three multilingual KG pairs extracted from DBpedia [1]: English to Chinese (`D-Z`), English to Japanese (`D-J`), and English to French (`D-F`); and **(2)** **`SRPRS`**, which is a sparser dataset that follows real-life entity distribution, including two multilingual KG pairs extracted from DBpedia: English to French (`S-F`) and English to German (`S-D`), and two mono-lingual KG pairs: DBpedia to Wikidata [53] (`S-W`) and DBpedia to YAGO [49] (`S-Y`); and **(3)** **`DWY100K`**, a larger dataset consisting of two mono-lingual KG pairs: DBpedia to Wikidata (`D-W`) and DBpedia to YAGO (`D-Y`). The detailed statistics can be found in Table 3, where the numbers of entities, relations, triples, gold links, and the average entity degree are reported. Regarding the gold alignment links, we adopted 70% as test set, 20% for training, and 10% for validation.

Evaluation metric. We utilized *F1 score* as the evaluation metric, which is the harmonic mean between *precision* and *recall*, where the *precision* value is computed as the number of correct matches divided by the number of matches found by a method, and the *recall* value is computed as the number of correct matches found by a method divided by the number of gold matches. Note that *recall* is equivalent to the Hits@1 metric used in some previous works.

Similarity metric. After obtaining the unified entity representations $\boldsymbol{E}$, a similarity metric is required to produce pairwise scores and generate the similarity matrix $\boldsymbol{S}$. Frequent choices include the cosine similarity [7], [36], [52], the Euclidean distance [8], [27] and the Manhattan distance [55], [58]. In this work, we followed mainstream works and adopted the cosine similarity.

Notably, we omit more detailed experimental settings in the interest of space, which can be found in Appendix B.

4.3 Main Results and Comparison

We first evaluate with only structural information and report the results in Table 4, where `R-` and `G-` refer to using RREA

5. The codes are publicly available at https://github.com/DexterZeng/EntMatcher

6. https://github.com/nju-websoft/OpenEA
7. https://github.com/THU-KEG/EAkit

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2023.3272584

TABLE 3
Dataset statistics.

	DBP15K			SRPRS				DWY100K		FB_DBP_MUL
	D-Z	D-J	D-F	S-F	S-D	S-W	S-Y	D-W	D-Y	
#Entities	38,960	39,594	39,654	30,000	30,000	30,000	30,000	200,000	200,000	44,716
#Relations	3,024	2,452	2,111	398	342	397	253	550	333	2,070
#Triples	165,556	170,698	221,720	70,040	75,740	78,580	70,317	912,068	931,515	164,882
#Gold links	15,000	15,000	15,000	15,000	15,000	15,000	15,000	100,000	100,000	22,117
Avg. degree	4.2	4.3	5.6	2.3	2.5	2.6	2.3	4.6	4.7	3.7

TABLE 4
The F1 scores of only using structural information.

	R-DBP				R-SRP					G-DBP				G-SRP				
	D-Z	D-J	D-F	Imp.	S-F	S-D	S-W	S-Y	Imp.	D-Z	D-J	D-F	Imp.	S-F	S-D	S-W	S-Y	Imp.
DInf	0.605	0.603	0.627		0.367	0.521	0.416	0.448		0.291	0.295	0.286		0.170	0.322	0.202	0.253	
CSLS	0.688	0.677	0.712	13.2%	0.406	0.550	0.465	0.481	8.6%	0.375	0.390	0.377	31.0%	0.224	0.368	0.258	0.306	22.1%
RInf	0.712	0.706	0.742	17.7%	0.412	0.560	0.477	0.486	10.4%	0.400	0.423	0.423	42.9%	0.241	0.381	0.276	0.324	29.0%
Sink.	**0.749**	0.740	**0.778**	23.5%	**0.423**	**0.568**	**0.480**	**0.497**	**12.3%**	0.447	0.471	**0.484**	60.8%	**0.248**	**0.387**	**0.289**	**0.331**	**32.5%**
Hun.	**0.749**	**0.744**	0.777	**23.7%**	0.418	0.563	0.475	0.495	11.4%	**0.450**	**0.480**	**0.484**	**62.2%**	0.246	0.385	0.284	**0.331**	31.6%
SMat	0.686	0.677	0.718	13.4%	0.398	0.551	0.453	0.471	6.9%	0.382	0.413	0.388	35.7%	0.231	0.371	0.260	0.312	24.0%
RL	0.675	0.670	0.716	12.3%	0.380	0.541	0.444	0.462	4.2%	0.378	0.409	0.371	32.9%	0.213	0.361	0.245	0.288	16.9%

TABLE 5
The F1 scores of using auxiliary information.

	N-DBP				N-SRP			NR-DBP				NR-SRP		
	D-Z	D-J	D-F	Imp.	S-F	S-D	Imp.	D-Z	D-J	D-F	Imp.	S-F	S-D	Imp.
DInf	0.735	0.780	0.744		0.815	0.831		0.819	0.862	0.846		0.865	0.893	
CSLS	0.754	0.802	0.761	2.6%	0.837	0.855	2.8%	0.858	0.896	0.880	4.2%	0.911	0.932	4.8%
RInf	0.751	0.802	0.761	2.4%	0.840	0.861	3.3%	0.861	0.899	0.887	4.7%	0.922	0.937	5.7%
Sink.	0.770	0.823	0.788	5.4%	0.853	0.878	5.2%	0.902	0.929	0.933	9.4%	0.940	0.954	7.7%
Hun.	**0.773**	**0.830**	**0.797**	**6.2%**	**0.864**	**0.887**	**6.4%**	**0.908**	**0.937**	**0.944**	**10.4%**	**0.949**	**0.956**	**8.4%**
SMat	0.768	0.818	0.778	4.6%	0.856	0.873	5.0%	0.879	0.912	0.906	6.7%	0.921	0.939	5.8%
RL	0.770	0.824	0.783	5.2%	0.851	0.866	4.3%	0.880	0.909	0.904	6.6%	0.917	0.936	5.4%

and GCN to generate the structural embeddings, respectively, DBP and SRP denote DBP15K and SRPRS, respectively. Next, we supplement with name embeddings, and report the results in Table 5, where N- and NR- refer to only using the name embeddings and fusing name embeddings with RREA structural representations, respectively. Note that, on existing datasets, all the entities in the test set can be matched, and all the algorithms are devised to find a target entity for each test source entity. Hence, the number of matches found by a method equals to the number of gold matches, and consequently the precision value is equal to the recall value and the F1 score [65].

Overall performance. First, we do not delve into the embedding matching algorithms and directly analyze the general results. Specifically, using RREA to learn structural representations can bring better performance compared with using GCN, showcasing that representation learning strategies are crucial to the overall alignment performance. When introducing the entity name information, it observes that this auxiliary signal alone can already provide very accurate signal for alignment. This is because the equivalent entities in different KGs of current datasets share very similar or even identical names. After fusing the semantic and structural information, the alignment performance is further lifted, with most of the approaches hitting over 0.9 in terms of the F1 score.

Effectiveness comparison of embedding matching algorithms. From the tables, it is evident that: (1) *Overall, Hun. and Sink. attain much better results than the other strategies.* Specifically, Hun. takes full account of the global matching constraints and strives to reach a globally optimal matching given the objective of maximizing the sum of pairwise similarity scores. Moreover, the 1-to-1 constraint it exerts aligns with present evaluation setting where the source and target entities are 1-to-1 matched. Sink., on the other hand, implicitly implements the 1-to-1 constraint during pairwise score computation and still adopts Greedy to produce final results, where there might exist non 1-to-1 matches; (2) *DInf attains the worst performance.* This is because it directly adopts the similarity scores that suffer from the hubness and isolation issues [50]. Besides, it leverages Greedy, which merely reaches the local optimum for each entity. (3) *The performance of RInf, CSLS, SMat and RL are well matched.* RInf and CSLS improve upon DInf by mitigating the hubness issue and enhancing the quality of pairwise scores. SMat and RL, on the other hand, improve upon DInf by modeling the interactions among matching decisions for different entities.

Furthermore, we conduct a deeper analysis of these approaches, and identify the following patterns:

Pattern 1. If for source entities, their highest pairwise similarity scores are close, RInf and CSLS (resp., SMat and RL) would attain relatively better (resp., worse) performance. Specifically, in Table 4 where RInf consistently (CSLS sometimes) attains superior results than SMat and RL, the average standard

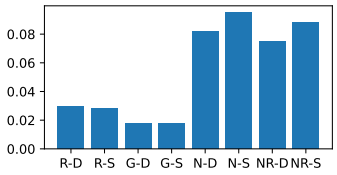
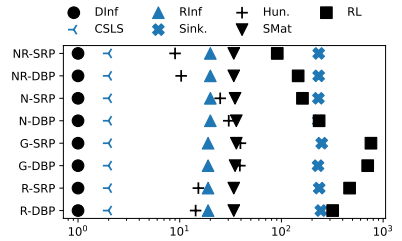
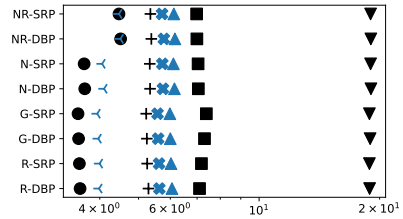


Fig. 4. The statistic of pairwise similarity scores (i.e., Top-5 STD).



(a) Time cost (in seconds).



(b) Memory space cost (in GB).

Fig. 5. Efficiency comparison. Shapes in blue denote methods that improve pairwise scores, while shapes in black denote those exerting global constraints (except for DInf).

deviation (STD) values of the top-5 pairwise similarity scores of source entities (cf. Figure 4) are very small, unveiling that the top scores are close and difficult to differentiate. In contrast, in Table 5 where SMat and RL outperform RInf and CSLS, the corresponding STD values are relatively large. This is because RInf and CSLS aim to make the scores more distinguishable, and hence they are more effective in cases where the top similarity scores are very close (i.e., low STD values). On the contrary, when the top similarity scores are already discriminating (e.g., Table 5), RInf and CSLS become less useful, while SMat and RL can still make improvements by using the global constraints to enforce the deviation from local optimums.

Pattern 2. On sparser datasets, the superiority of Sink. and Hun. over the rest of the methods becomes less significant. This is based on the observation that on SRPRS, other matching algorithms (RInf in particular) attain much closer performance to Sink. and Hun.. Such a pattern could be attributed to the fact that, on sparser datasets, entities normally have fewer connections with others, i.e., lower average entity degree (in Table 3), where representation learning strategies might fail to fully capture the structural signals for alignment and the resultant pairwise scores become less accurate. These inaccurate scores could mislead the matching process and hence limit the effectiveness of the top-performing methods, i.e., Sink. and Hun.. In other words, sparser KG structures are more likely to (partially) break the fundamental assumption on KG structure similarity (cf. Section 2.3).

Efficiency analysis. We compare the time and space efficiency of these methods on the medium-sized datasets in Figure 5. Since the costs on KG pairs from the same dataset are very similar, we report the *average* time and space costs under each setting in the interest of space.

Specifically, it observes that: (1) The simple algorithm DInf is the most efficient approach; (2) *Among the advanced approaches, CSLS is the most efficient one*, closely following DInf; (3) *The efficiency of RInf and Hun. are equally matched*. While Hun. consumes relatively less memory space than RInf, its time efficiency is less stable and tends to run slower on datasets with less accurate pairwise scores; (4) The space efficiency of Sink. is close to RInf and Hun., whereas it has much higher time costs, which largely depends on the value of l; (5) *RL is the least time-efficient approach, while SMat is the least space-efficient algorithm*. RL requires more time on datasets with less accurate pairwise scores where its pre-processing module fails to produce promising results [65]. The memory space consumption of SMat is high, as it needs to store a large amount of intermediate matching results. In all, we can conclude that generally, advanced embedding matching algorithms require more time and memory space, among which the methods incorporating global matching constraints tend to be less efficient.

Comparison with DL-based EM approaches. We utilize the deepmatcher python package [39], which provides built-in neural networks and utilities that can train and apply state-of-the-art deep learning models for entity matching, to address EA. Specifically, we use the structural and name embeddings to replace the attributive text inputs in deepmatcher, respectively, and then train the neural model with labeled data. For each positive entity pair, we randomly sample 10 negative ones. In the testing stage, for each source entity, we feed the entity pairs constituting it and all the target entities into the trained classifier, and regard the entity pair with the highest predicted score as the result.

In the final results, only several entities are correctly aligned, showing that DL-based EM approaches cannot handle EA, which can be ascribed to the insufficient labeled data, imbalanced class distribution and the lack of attributive text information, as discussed in Section 2.2.

4.4 Results on Large-scale Datasets

Next, we provide the results on the relatively larger dataset, i.e., DWY100K, which can also reflect the scalability of these algorithms. The results are presented in Table 6 [8]. The general pattern is similar to that on G-DBP (i.e., using GCN on DBP15K), where Sink. and Hun. obtain the best results, followed by RInf. The performance of CSLS and RL are close, outperforming DInf by over 20%.

We compare the efficiency of these algorithms in Table 6, where $\bar{T}$ refers to the average time cost and Mem. denotes whether the memory space required by the model can be covered by our experimental environment [9]. It observes that, given larger datasets, most of the performant algorithms have poor efficiency and scalability (e.g., RInf, Sink. and Hun.). Note that in [62], two variants of RInf, i.e., RInf-wr and RInf-pb,

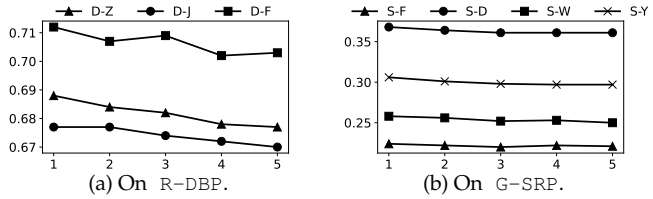
8. We cannot provide the results of SMat, as it requires extremely large memory space and cannot work under our experimental environment.

9. Note that for algorithms with memory space costs exceeding our experimental environment (except for SMat), there is additional swap area in the hard drive for them to finish the program (which usually takes much longer time).

TABLE 6
The F1 scores on DWY100K using GCN.

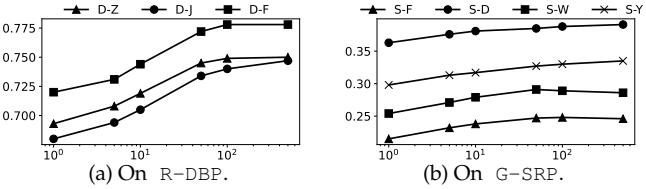
	D-W	D-Y	Imp.	$\bar{T}$	Mem.
DInf	0.409	0.552		**4**	Yes
CSLS	0.510	0.650	20.7%	83	Yes
RInf	0.559	0.692	30.2%	1,102	No
RInf-wr	0.510	0.650	20.7%	28	Yes
RInf-pb	0.524	0.663	23.5%	289	Yes
Sink.	**0.618**	**0.739**	**41.2%**	9,405	No
Hun.	**0.618**	0.734	40.6%	3,607	No
SMat	/	/	/	/	/
RL	0.520	0.660	22.8%	995	Yes

are proposed to improve its scalability at the cost of a small performance drop, which is empirically validated in Table 6. This also reveals that more scalable matching algorithms for KGs in entity embedding spaces should be devised.



(a) On R-DBP. (b) On G-SRP.

Fig. 6. F1 scores of CSLS with varying k value.



(a) On R-DBP. (b) On G-SRP.

Fig. 7. F1 scores of Sink. with varying l value.

4.5 Analysis and Insights

We provide further experiments and discussions in this subsection. Due to the limitation of space, more experiments and the case study can be found in Appendix C and Appendix D.

On efficiency and scalability. The simple algorithm DInf is the most efficient and scalable one, as it merely involves the most basic computation and matching operations. CSLS is slightly less efficient than DInf due to the update of pairwise similarity scores. It also has good scalability. Although RInf adopts a similar idea to CSLS, it involves an additional ranking process, which brings much more time and memory consumption, making it less scalable. Sink. repeatedly conducts the normalization operation, and thus its time efficiency is mainly up to the l value. Its scalability is also limited by the memory space consumption since it needs to store intermediate results, as revealed in Table 6.

Regarding the methods that exert global constraints, Hun. is efficient on medium-sized datasets, while it is not scalable due to the high time complexity and memory space consumption. SMat is space-inefficient even on the medium-sized datasets, making it not scalable. In comparison, RL has more stable time and space costs and can scale to large datasets, and the main influencing factor is the accuracy of pairwise scores. This is because RL has a pre-processing step that filters out confident matched entity pairs and excludes them from the time-consuming RL learning process [65]. More confident matched entity pairs would be filtered out if the pairwise scores are more accurate.

On effectiveness of improving pairwise score computation. We compare and discuss the strategies for improving the pairwise score computation, i.e., CSLS, RInf and Sink..

Both CSLS and RInf aim to mitigate the hubness and isolation issues in the raw pairwise scores (from different starting points). Particularly, we observe that, by setting k (in Equation 1) of CSLS to 1, the difference between RInf and CSLS is reduced to the extra *ranking* process of RInf, and the results in Table 4 and 5 validate that this ranking process can consistently bring better performance. This is because the ranking operation can amplify the difference among the scores and prevent such information from being lost after the bidirectional aggregation [62]. However, it is noteworthy that the ranking process brings much more time and memory consumption, as can be observed from the empirical results.

Then we analyze the influence of k value in CSLS. As shown in Figure 6, a larger k leads to worse performance. This is because a larger k implies a smaller ϕ value in Equation 1 (where the top-k highest scores are considered and averaged), and the resultant pairwise scores become less distinctive. This also validates the effectiveness of the design in RInf (cf. Equation 2), where only the *maximum* value is considered to compute the preference score. Nevertheless, in Section 5.2, we reveal that setting k to 1 is only useful in the 1-to-1 alignment setting.

As for Sink., it adopts an extreme approach to optimize the pairwise scores, which encourages each source (resp., target) entity to *have only one positive pairwise score* with a target (resp., source) entity and 0's with the rest of the target (resp., source) entities. Thus, it is in fact progressively and implicitly implementing the 1-to-1 alignment constraint during the pairwise score computation process with the increase of l, and is particularly useful in present 1-to-1 evaluation settings of EA. In Figure 7, we further examine the influence of l in Equation 3 on the alignment results of Sink., which meets our expectation that the larger the l value, the better the distribution of the resultant pairwise scores fits the 1-to-1 constraint, and thus the higher the alignment performance. Nevertheless, a larger l also implies longer processing time. Therefore, by tuning on the validation set, we set l to 100 to reach the balance between effectiveness and efficiency.

On effectiveness of exerting global constraints. Next, we compare and discuss the methods that exert global constraints on the embedding matching process, i.e., Hun., SMat and RL.

It is evident that Hun. is the most performant approach, as it fits well with the present EA setting and can secure an optimal solution towards maximizing the sum of pairwise scores. Specifically, the current EA setting has two notable assumptions (cf. Section 2.3). With these two assumptions, EA can be transformed into the linear assignment problem, which aims to maximize the sum of pairwise scores under the 1-to-1 constraint [35]. As thus, the algorithms for solving the linear assignment problem, e.g., Hun., can attain remarkably high performance on EA. However, these two assumptions do not

necessarily hold on all occasions, which could influence the effectiveness of Hun.. For instance, as revealed in Pattern 2, on sparse datasets (e.g., SRPRS), the neighboring structures of some equivalent entities are likely to be different, where the effectiveness of Hun. is limited. In addition, the 1-to-1 alignment constraint is not necessarily true in practice, which will be discussed in Section 5.

In comparison, SMat merely aims to attain a stable matching, where the resultant entity pairing could be sub-optimal under present evaluation setting. RL, on the other hand, relaxes the 1-to-1 constraint and only deviates slightly from the greedy matching, and hence the results are not very promising.

Overall comparison and conclusion. Finally, we compare the algorithms all together and draw the following conclusions *under the 1-to-1 alignment setting*: (1) The best performing methods are Hun. and Sink.. Nevertheless, they have low scalability; (2) CSLS and RInf achieve the best balance between effectiveness and efficiency. While CSLS is more efficient, RInf is more effective; (3) SMat and RL tend to attain better results when the accuracy of the pairwise scores is high. Nevertheless, they require relatively more time.

5 New Evaluation Settings

In this section, we conduct experiments on settings that can better reflect real-life challenges.

5.1 Unmatchable Entities

Current EA literature largely overlooks the unmatchable issue, where a KG contains entities that the other KG does not contain. For instance, when aligning YAGO 4 and IMDB, only 1% of entities in YAGO 4 are film-related and possibly have equivalent entities in IMDB, while the other 99% of entities in YAGO 4 necessarily have no match in IMDB [68]. Hence, we aim to evaluate the embedding matching algorithms in terms of dealing with unmatchable entities.

Datasets and evaluation settings. Following [63], we adapt the KG pairs in DBP15K to include unmatchable entities, resulting in DBP15K+. More specific construction procedure can be found in [63]. As for the evaluation metric, we follow the main experimental setting and adopt the *F1 score*. Unlike 1-to-1 alignment, there exist unmatchable entities in this adapted dataset, and the precision and recall values are not necessarily equivalent, since some methods would also align unmatchable entities. Noteworthily, the original setting of SMat and Hun. requires that the numbers of entities on the two sides are equal. Thus, we add the dummy nodes on the side with fewer entities to restore such a setting, and then apply SMat and Hun.. The corresponding results are reported in Table 7.

Alignment results. It reads that Hun. attains the best results, followed by SMat. The superior results are partially due to the addition of dummy nodes, which could mitigate the unmatchable issue to a certain degree. The results RInf and Sink. are close, outperforming CSLS and RL. DInf still achieves the worst performance.

Besides, by comparing the results on DBP15K+ and those on the original dataset DBP15K (cf. Table 4), we observe that: (1) After including the unmatchable entities, for all methods, the F1 scores drop. This is because most of current embedding matching algorithms are greedy, i.e., retrieving a target entity for each source entity (including the unmatchable ones), which leads to a very low precision. For the rest of the methods, e.g., Hun. and SMat, the unmatchable entities also mislead the matching process and thus affect the final results; (2) Unlike on DBP15K where the performance of Sink. and Hun. are close, on DBP15K+, Hun. largely outperforms Sink., as Hun. does not necessarily align a target entity to each source entity and has a higher precision; (3) Overall, existing algorithms for matching KGs in entity embedding spaces lack the capability of dealing with unmatchable entities.

TABLE 7
F1 scores on DBP15K+.

	GCN				RREA			
	D-Z	D-J	D-F	$\bar{T}$	D-Z	D-J	D-F	$\bar{T}$
DInf	0.241	0.240	0.234	**1**	0.501	0.491	0.513	**1**
CSLS	0.310	0.318	0.309	2	0.569	0.551	0.582	2
RInf	0.333	0.344	0.344	28	0.582	0.568	0.599	28
Sink.	0.329	0.337	0.343	336	0.571	0.553	0.584	331
Hun.	**0.397**	**0.407**	**0.408**	115	**0.712**	**0.706**	**0.750**	46
SMat	0.366	0.386	0.367	140	0.673	0.665	0.707	144
RL	0.307	0.311	0.297	1738	0.553	0.531	0.579	1264

5.2 Non 1-to-1 Alignment

Next, we study the setting where the source and target entities do not strictly conform to the 1-to-1 constraint, so as to better appreciate these matching algorithms for KGs in entity embedding spaces. Non 1-to-1 alignment is common in practice, especially when two KGs contain entities in different granularity, or one KG is noisy and involves duplicate entities. *To the best of our knowledge, we are among the first attempts to identify and investigate this issue.*

Dataset construction. Present EA benchmarks are constructed according to the 1-to-1 constraint. Thus, in this work, we establish a new dataset that involves non 1-to-1 alignment relationships. Specifically, we obtain the pre-annotated links [10] between Freebase [3] and DBpedia [1], and preserve the entities that are involved in 1-to-many, many-to-1, and many-to-many alignment relationships. Then, we retrieve the relational triples that contain these entities from respective KGs, which also introduces new entities. Next, we detect the links among the newly added entities, and add them into the alignment links. Finally, the resultant dataset, FB_DBP_MUL, contains 44,716 entities, 164,882 triples, 22,117 gold links, among which 20,353 are non 1-to-1 links and 1,764 are 1-to-1 links [11]. The specific statistics are also presented in Table 3.

Evaluation settings. To keep the integrity of the links among entities, we sample the training, validation and test sets from the gold links according to the principle that the links involving the same entity should not be distributed among different sets. The size of the final training, validation and test sets is approximately 7:1:2. We compare the entity pairs produced by embedding matching algorithms against the gold test links, and report the precision (P), recall (R) and F1 values.

Alignment results. It is evident from Table 8 that, compared with 1-to-1 alignment, the results change significantly on the

10. https://www.dbpedia.org/blog/dbpedia-is-now-interlinked-with-freebase-links-to-opencyc-updated/

11. FB_DBP_MUL is publicly available at https://github.com/DexterZeng/EntMatcher .

TABLE 8
The results on non 1-to-1 alignment dataset.

	GCN				RREA			
	P	R	F1	T	P	R	F1	T
DInf	0.074	0.051	0.061	**11**	0.167	0.114	0.136	**12**
CSLS	0.091	0.062	0.074	13	0.189	**0.130**	0.154	15
RInf	**0.093**	**0.064**	**0.076**	35	**0.190**	**0.130**	**0.155**	35
Sink.	0.083	0.057	0.068	286	0.180	0.124	0.147	278
Hun.	0.079	0.054	0.064	44	0.176	0.121	0.143	44
SMat	0.071	0.048	0.057	43	0.162	0.111	0.132	41
RL	0.066	0.045	0.054	1710	0.150	0.103	0.122	1440

new dataset. Specifically: (1) RInf and CSLS attain the best F1 scores, whereas the results are not very promising (e.g., with F1 score lower than 0.1 when using GCN); (2) Sink. and Hun. achieve much worse results compared with the performance on 1-to-1 alignment datasets; (3) The results of SMat and RL are even inferior to those of the simple baseline DInf. The main reason accounting for these changes is that the non 1-to-1 alignment links pose great challenges to existing embedding matching algorithms. Specifically, for DInf, CSLS, RInf, Sink. and RL, they only align *one* target entity (that possesses the highest score) to a given source entity, but fail to discover other alignment links that also involve this source entity. For SMat and Hun., they impose the 1-to-1 constraint during matching, which falls short on the non 1-to-1 setting, thus leading to inferior results. Therefore, it calls for the study on embedding matching algorithms targeted at non 1-to-1 alignment. We also discuss the k value in CSLS and RInf under the non 1-to-1 setting, which can be found in Appendix C.

6 Summary and Future Direction

In this section, we summarize the observations and insights made from our evaluation, and provide possible future research directions.

(1) The investigation into matching KGs in embedding spaces has not yet made substantial progress. Although there are a few algorithms tailored for matching KGs in embedding spaces, e.g., CSLS, RInf and RL, under the most popular EA evaluation setting (with 1-to-1 alignment constraint), they are outperformed by the classic general matching algorithms, i.e., Hun.. Hence, there is still much room for improving matching KGs in embedding spaces.

(2) No existing embedding matching algorithm prevails under all experimental settings. The strategies designed to solve the linear assignment problem attain the best performance under the 1-to-1 setting, while they fall short on more practical and challenging scenarios since the new settings (e.g., non 1-to-1 alignment) no longer align with the conditions of these optimization algorithms. Similarly, although the methods for improving the computation of pairwise scores achieve superior results in the non 1-to-1 alignment scenario, they are outperformed by other solutions under the unmatchable setting. Therefore, each evaluation setting poses its own challenge to the embedding matching process, and currently there is no consistent winner.

(3) The adaptation from general matching algorithms requires careful design. Among the embedding matching algorithms, Hun. and SMat are general matching algorithms that have been applied to many other related tasks. Although directly adopting these general strategies to tackle EA is simple and effective, they might well fall short in some scenarios, as the alignment on KGs possesses it own challenges, e.g., the matching is not necessarily 1-to-1 constrained, or the pairwise scores are inaccurate. Thus, it is suggested to take full account of the characteristics of the alignment settings when adapting other general matching algorithms to cope with matching KGs in entity embedding spaces.

(4) The scalability and efficiency should be brought to the attention. Existing advanced embedding matching algorithms have poor scalability, due to the additional resource-consuming operations that contribute to the alignment performance, such as the ranking process in RInf and the 1-to-1 constraint exerted by Hun. and SMat. Besides, the space efficiency is also a critical issue. As shown in Section 4.4, most of the approaches have rather high memory costs given large-scale datasets. Therefore, considering that in practice there are much more entities, the scalability and efficiency issues should be considered during the algorithm design. A preliminary exploration has been conducted by [15].

(5) The practical evaluation settings are worth further investigation. Under the unmatchable and non 1-to-1 alignment settings, the performance of existing algorithms is not promising. A possible future direction is to introduce the notion of probability and leverage the probabilistic reasoning frameworks [22], [45], which have higher flexibility, to produce the alignment results.

(6) Integrating the relation embedding might help. Two latest studies propose to use relation embeddings to help induce aligned entity pairs [33], [56]. Different from existing methods that regard EA as a matrix (second-order tensor) isomorphism problem, they express the isomorphism of KGs in the form of third-order tensors to better describe the structural information of KGs [33]. Thus, it might be interesting to study the matching between KGs in *joint* entity and relation embedding space.

We also provide some actionable insights:

1. In 1-to-1 constrained scenarios, it is preferable to use Hungarian algorithm or the Sinkhorn operation to conduct the matching, as they 1) implement the 1-to-1 constraint during execution, and 2) take full account of the global matching constraints and strive to reach a globally optimal matching given the objective of maximizing the sum of pairwise similarity scores. Given large-scale datasets, using Hungarian algorithm would be more time-efficient, as Sinkhorn operation needs to operate for multiple rounds to achieve convergence. Besides, while Hungarian algorithm depends mainly on CPU, Sinkhorn operation relies on GPU.

2. Given datasets with unmatchable entities, it is suggested to add dummy nodes to make the number of entities in both sides equal, and then use the Hungarian algorithm. In this case, there is still much room for improvement.

3. Non 1-to-1 alignment is a realistic and frequently observed scenario that has not received much research attention. Among existing algorithms, RInf and CSLS are preferred, since they take into account the global influence on the local matching and meanwhile do not strictly enforce the 1-to-1constraint. More practical solutions are to be put forward to effectively address non 1-to-1 alignment.

4. Current best performing embedding matching algorithms are not scalable, and the Hungarian algorithm requires approximately one hour on the `DWY100K` dataset. Hence, in this case, it might be better to utilize the RInf and its variant algorithms, which saves 2/3 of time cost at the cost of $< 10\%$ performance drop compared with the Hungarian algorithm.

7 Conclusion

This paper conducts a comprehensive survey and evaluation of matching algorithms for KGs in entity embedding spaces. We evaluate seven state-of-the-art strategies in terms of effectiveness and efficiency on a wide range of datasets, including two experimental settings that better mirror real-life challenges. We identify the strengths and weaknesses of these algorithms under different settings. We hope the experimental results would be valuable for researchers to put forward more effective and scalable embedding matching algorithms.

References

[1] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. G. Ives. Dbpedia: A nucleus for a web of open data. In *ISWC*, pages 722–735, 2007.

[2] M. Berrendorf, E. Faerman, V. Melnychuk, V. Tresp, and T. Seidl. Knowledge graph entity alignment with graph convolutional networks: Lessons learned. In *ECIR*, volume 12036, pages 3–11, 2020.

[3] K. D. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, pages 1247–1250, 2008.

[4] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, pages 2787–2795, 2013.

[5] U. Brunner and K. Stockinger. Entity matching with transformer architectures - A step forward in data integration. In *Proceedings of the 23rd International Conference on Extending Database Technology, EDBT 2020, Copenhagen, Denmark, March 30 - April 02, 2020*, pages 463–473. OpenProceedings.org, 2020.

[6] W. Cai, W. Ma, J. Zhan, and Y. Jiang. Entity alignment with reliable path reasoning and relation-aware heterogeneous graph transformer. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 1930–1937. ijcai.org, 2022.

[7] Y. Cao, Z. Liu, C. Li, Z. Liu, J. Li, and T. Chua. Multi-channel graph neural network for entity alignment. In *ACL*, pages 1452–1461, 2019.

[8] M. Chen, Y. Tian, K. Chang, S. Skiena, and C. Zaniolo. Co-training embeddings of knowledge graphs and entity descriptions for cross-lingual entity alignment. In *IJCAI*, pages 3998–4004, 2018.

[9] M. Chen, Y. Tian, M. Yang, and C. Zaniolo. Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. In *IJCAI*, pages 1511–1517, 2017.

[10] V. Christophides, V. Efthymiou, T. Palpanas, G. Papadakis, and K. Stefanidis. An overview of end-to-end entity resolution for big data. *ACM Comput. Surv.*, 53(6):127:1–127:42, 2021.

[11] A. Doan, P. Konda, P. S. G. C., Y. Govind, D. Paulsen, K. Chandrasekhar, P. Martinkus, and M. Christie. Magellan: toward building ecosystems of entity matching solutions. *Commun. ACM*, 63(8):83–91, 2020.

[12] J. Doerner, D. Evans, and A. Shelat. Secure stable matching at scale. In *SIGSAC Conference*, pages 1602–1613, 2016.

[13] M. Fey, J. E. Lenssen, C. Morris, J. Masci, and N. M. Kriege. Deep graph matching consensus. In *ICLR*, 2020.

[14] D. Gale and L. S. Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

[15] Y. Gao, X. Liu, J. Wu, T. Li, P. Wang, and L. Chen. Clusterea: Scalable entity alignment with stochastic training and normalized mini-batch similarities. In *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, pages 421–431. ACM, 2022.

[16] C. Ge, X. Liu, L. Chen, B. Zheng, and Y. Gao. Largeea: Aligning entities for large-scale knowledge graphs. *CoRR*, abs/2108.05211, 2021.

[17] C. Ge, X. Liu, L. Chen, B. Zheng, and Y. Gao. Make it easy: An effective end-to-end entity alignment framework. In *SIGIR*, pages 777–786, 2021.

[18] C. Ge, P. Wang, L. Chen, X. Liu, B. Zheng, and Y. Gao. Collaborem: A self-supervised entity matching framework using multi-features collaboration. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2021.

[19] L. Guo, Q. Zhang, Z. Sun, M. Chen, W. Hu, and H. Chen. Understanding and improving knowledge graph embedding for entity alignment. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 8145–8156. PMLR, 2022.

[20] E. Jiménez-Ruiz and B. C. Grau. Logmap: Logic-based and scalable ontology matching. In *ISWC*, pages 273–288. Springer, 2011.

[21] R. Jonker and A. Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38(4):325–340, 1987.

[22] A. Kimmig, A. Memory, R. J. Miller, and L. Getoor. A collective, probabilistic approach to schema mapping. In *ICDE*, pages 921–932, 2017.

[23] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*. OpenReview.net, 2017.

[24] H. W. Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[25] S. Lacoste-Julien, K. Palla, A. Davies, G. Kasneci, T. Graepel, and Z. Ghahramani. Sigma: simple greedy matching for aligning large knowledge bases. In *SIGKDD*, pages 572–580. ACM, 2013.

[26] G. Lample, A. Conneau, M. Ranzato, L. Denoyer, and H. Jégou. Word translation without parallel data. In *ICLR*, 2018.

[27] C. Li, Y. Cao, L. Hou, J. Shi, J. Li, and T. Chua. Semi-supervised entity alignment via joint knowledge embedding model and cross-graph model. In *EMNLP*, pages 2723–2732, 2019.

[28] J. Li and D. Song. Uncertainty-aware pseudo label refinery for entity alignment. In *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*, pages 829–837. ACM, 2022.

[29] Y. Li, J. Li, Y. Suhara, A. Doan, and W. Tan. Deep entity matching with pre-trained language models. *Proc. VLDB Endow.*, 14(1):50–60, 2020.

[30] X. Lin, H. Yang, J. Wu, C. Zhou, and B. Wang. Guiding cross-lingual entity alignment via adversarial knowledge embedding. In *ICDM*, pages 429–438, 2019.

[31] B. Liu, H. Scells, G. Zuccon, W. Hua, and G. Zhao. Activeea: Active learning for neural entity alignment. In *EMNLP*, pages 3364–3374, 2021.

[32] X. Liu, H. Hong, X. Wang, Z. Chen, E. Kharlamov, Y. Dong, and J. Tang. Selfkg: Self-supervised entity alignment in knowledge graphs. In *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*, pages 860–870. ACM, 2022.

[33] X. Mao, M. Ma, H. Yuan, J. Zhu, Z. Wang, R. Xie, W. Wu, and M. Lan. An effective and efficient entity alignment decoding algorithm via third-order tensor isomorphism. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 5888–5898. Association for Computational Linguistics, 2022.

[34] X. Mao, W. Wang, Y. Wu, and M. Lan. Boosting the speed of entity alignment 10 ×: Dual attention matching network with normalized hard sample mining. In *WWW*, pages 821–832, 2021.

[35] X. Mao, W. Wang, Y. Wu, and M. Lan. From alignment to assignment: Frustratingly simple unsupervised entity alignment. In *EMNLP*, pages 2843–2853, 2021.

[36] X. Mao, W. Wang, H. Xu, Y. Wu, and M. Lan. Relational reflection entity alignment. In *CIKM*, pages 1095–1104, 2020.

[37] G. E. Mena, D. Belanger, S. W. Linderman, and J. Snoek. Learning latent permutations with gumbel-sinkhorn networks. In *ICLR*, 2018.

[38] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML*, volume 48, pages 1928–1937, 2016.

[39] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, and V. Raghavendra. Deep learning for entity matching: A design space exploration. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, pages 19–34. ACM, 2018.

[40] T. T. Nguyen, T. T. Huynh, H. Yin, V. V. Tong, D. Sakong, B. Zheng, and Q. V. H. Nguyen. Entity alignment for knowledge graphs with multi-order convolutional networks. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2020.

[41] G. Papadakis, D. Skoutas, E. Thanos, and T. Palpanas. Blocking and filtering techniques for entity resolution: A survey. *ACM Comput. Surv.*, 53(2):31:1–31:42, 2020.
[42] H. Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web*, 8(3):489–508, 2017.
[43] S. Pei, L. Yu, and X. Zhang. Improving cross-lingual entity alignment via optimal transport. In S. Kraus, editor, *IJCAI*, pages 3231–3237, 2019.
[44] L. A. S. Pizzato, T. Rej, T. Chung, I. Koprinska, and J. Kay. RECON: a reciprocal recommender for online dating. In *RecSys*, pages 207–214. ACM, 2010.
[45] J. Pujara, H. Miao, L. Getoor, and W. W. Cohen. Large-scale knowledge graph identification using PSL. In *AAAI*, 2013.
[46] A. E. Roth. Deferred acceptance algorithms: history, theory, practice, and open questions. *Int. J. Game Theory*, 36(3-4):537–569, 2008.
[47] P. Shvaiko and J. Euzenat. Ontology matching: State of the art and future challenges. *IEEE Trans. Knowl. Data Eng.*, 25(1):158–176, 2013.
[48] F. M. Suchanek, S. Abiteboul, and P. Senellart. PARIS: probabilistic alignment of relations, instances, and schema. *Proc. VLDB Endow.*, 5(3):157–168, 2011.
[49] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *WWW*, pages 697–706, 2007.
[50] Z. Sun, Q. Zhang, W. Hu, C. Wang, M. Chen, F. Akrami, and C. Li. A benchmarking study of embedding-based entity alignment for knowledge graphs. *Proc. VLDB Endow.*, 13(11):2326–2340, 2020.
[51] A. Surisetty, D. Chaurasiya, N. Kumar, A. Singh, G. Dhama, A. Malhotra, A. Arora, and V. Dey. Reps: Relation, position and structure aware entity alignment. In *Companion of The Web Conference 2022, Virtual Event / Lyon, France, April 25 - 29, 2022*, pages 1083–1091. ACM, 2022.
[52] B. D. Trisedya, J. Qi, and R. Zhang. Entity alignment between knowledge graphs using attribute embeddings. In *AAAI*, pages 297–304, 2019.
[53] D. Vrandecic and M. Krötzsch. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85, 2014.
[54] Z. Wang, Q. Lv, X. Lan, and Y. Zhang. Cross-lingual knowledge graph alignment via graph convolutional networks. In *EMNLP*, pages 349–357, 2018.
[55] Y. Wu, X. Liu, Y. Feng, Z. Wang, R. Yan, and D. Zhao. Relation-aware entity alignment for heterogeneous knowledge graphs. In *IJCAI*, pages 5278–5284, 2019.
[56] K. Xin, Z. Sun, W. Hua, W. Hu, and X. Zhou. Informed multi-context entity alignment. In *WSDM '22: The Fifteenth ACM International Conference on Web Search and Data Mining, Virtual Event / Tempe, AZ, USA, February 21 - 25, 2022*, pages 1197–1205. ACM, 2022.
[57] K. Xu, L. Song, Y. Feng, Y. Song, and D. Yu. Coordinated reasoning for cross-lingual knowledge graph alignment. In *AAAI*, pages 9354–9361, 2020.
[58] H. Yang, Y. Zou, P. Shi, W. Lu, J. Lin, and X. Sun. Aligning cross-lingual entities with multi-aspect information. In *EMNLP*, pages 4430–4440, 2019.
[59] J. Yang, D. Wang, W. Zhou, W. Qian, X. Wang, J. Han, and S. Hu. Entity and relation matching consensus for entity alignment. In *CIKM*, pages 2331–2341. ACM, 2021.
[60] K. Zeng, Z. Dong, L. Hou, Y. Cao, M. Hu, J. Yu, X. Lv, L. Cao, X. Wang, H. Liu, Y. Huang, J. Feng, J. Wan, J. Li, and L. Feng. Interactive contrastive learning for self-supervised entity alignment. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management, Atlanta, GA, USA, October 17-21, 2022*, pages 2465–2475. ACM, 2022.
[61] K. Zeng, C. Li, L. Hou, J. Li, and L. Feng. A comprehensive survey of entity alignment for knowledge graphs. *AI Open*, 2:1–13, 2021.
[62] W. Zeng, X. Zhao, X. Li, J. Tang, and W. Wang. On entity alignment at scale. *VLDB J.*, page in press, 2021.
[63] W. Zeng, X. Zhao, J. Tang, X. Li, M. Luo, and Q. Zheng. Towards entity alignment in the open world: An unsupervised approach. In *DASFAA 2021*, volume 12681, pages 272–289, 2021.
[64] W. Zeng, X. Zhao, J. Tang, and X. Lin. Collective entity alignment via adaptive features. In *ICDE*, pages 1870–1873. IEEE, 2020.
[65] W. Zeng, X. Zhao, J. Tang, X. Lin, and P. Groth. Reinforcement learning-based collective entity alignment with adaptive features. *ACM Trans. Inf. Syst.*, 39(3):26:1–26:31, 2021.
[66] R. Zhang, B. D. Trisedya, M. Li, Y. Jiang, and J. Qi. A benchmark and comprehensive survey on knowledge graph entity alignment via representation learning. *VLDB J.*, 31(5):1143–1168, 2022.
[67] Z. Zhang, H. Liu, J. Chen, X. Chen, B. Liu, Y. Xiang, and Y. Zheng. An industry evaluation of embedding-based entity alignment. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020 - Industry Track, Online, December 12, 2020*, pages 179–189. International Committee on Computational Linguistics, 2020.
[68] X. Zhao, W. Zeng, J. Tang, W. Wang, and F. Suchanek. An experimental study of state-of-the-art entity alignment approaches. *IEEE TKDE*, pages 1–1, 2020.
[69] R. Zhu, M. Ma, and P. Wang. RAGA: relation-aware graph attention networks for global entity alignment. In *PAKDD*, volume 12712, pages 501–513, 2021.
[70] Y. Zhu, H. Liu, Z. Wu, and Y. Du. Relation-aware neighborhood matching model for entity alignment. In *AAAI*, pages 4749–4756, 2021.

Weixin Zeng received the PhD degree from National University of Defense Technology (NUDT), China, in 2022. He is a lecturer with NUDT, and his research mainly focuses on knowledge graphs and data mining.

Xiang Zhao received the PhD degree from The University of New South Wales, Australia, in 2014. He is currently a professor at National University of Defense Technology, China. His research interests include graph data management and mining, with a special focus on knowledge graphs.

Zhen Tan received the PhD degree from National University of Defense Technology (NUDT), China, in 2018. He is currently an associate professor with NUDT. His research interests include knowledge graphs and advanced data analytics.

Jiuyang Tang received the PhD degree from National University of Defense Technology (NUDT), China, in 2004. He is currently a professor with NUDT. His research interests include knowledge graphs and advanced data analytics.

Xueqi Cheng is currently a Professor with the Institute of Computing Technology, Chinese Academy of Sciences. His research interests include network science, web search and data mining, big data processing, and distributed computing architecture.