Domain Adaptation for Deep Entity Resolution

Jianhong Tu
Renmin University, China
tujh@ruc.edu.cn

Ju Fan*
Renmin University, China
fanj@ruc.edu.cn

Nan Tang
QCRI, Qatar
ntang@hbku.edu.qa

Peng Wang
Renmin University, China
lisa_wang@ruc.edu.cn

Chengliang Chai
Tsinghua University, China
ccl@mail.tsinghua.edu.cn

Guoliang Li
Tsinghua University, China
liguoliang@tsinghua.edu.cn

Ruixue Fan
Renmin University, China
fanruixue@ruc.edu.cn

Xiaoyong Du
Renmin University, China
duyong@ruc.edu.cn

ABSTRACT

Entity resolution (ER) is a core problem of data integration. The state-of-the-art (SOTA) results on ER are achieved by deep learning (DL) based methods, trained with a lot of labeled matching/non-matching entity pairs. This may not be a problem when using well-prepared benchmark datasets. Nevertheless, for many real-world ER applications, the situation changes dramatically, with a painful issue to collect large-scale labeled datasets. In this paper, we seek to answer: *If we have a well-labeled source ER dataset, can we train a DL-based ER model for a target dataset, without any labels or with a few labels?* This is known as *domain adaptation* (DA), which has achieved great successes in computer vision and natural language processing, but is not systematically studied for ER. Our goal is to systematically explore the benefits and limitations of a wide range of DA methods for ER. To this purpose, we develop a **DADER** (**D**omain **A**daptation for **D**eep **E**ntity **R**esolution) framework that significantly advances ER in applying DA. We define a space of design solutions for the three modules of **DADER**, namely *Feature Extractor*, *Matcher*, and *Feature Aligner*. We conduct so far the most comprehensive experimental study to explore the design space and compare different choices of DA for ER. We provide guidance for selecting appropriate design solutions based on extensive experiments.

CCS CONCEPTS

• **Information systems** → **Entity resolution**.

KEYWORDS

Entity Resolution; Domain Adaptation; Deep Learning

ACM Reference Format:
Jianhong Tu, Ju Fan, Nan Tang, Peng Wang, Chengliang Chai, Guoliang Li, Ruixue Fan, and Xiaoyong Du. 2022. Domain Adaptation for Deep Entity Resolution. In *Proceedings of the 2022 International Conference on Management of Data (SIGMOD '22), June 12–17, 2022, Philadelphia, PA, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3514221.3517870

*Ju Fan is the corresponding author.

SIGMOD '22, June 12–17, 2022, Philadelphia, PA, USA.

ACM ISBN 978-1-4503-9249-5/22/06…$15.00
https://doi.org/10.1145/3514221.3517870

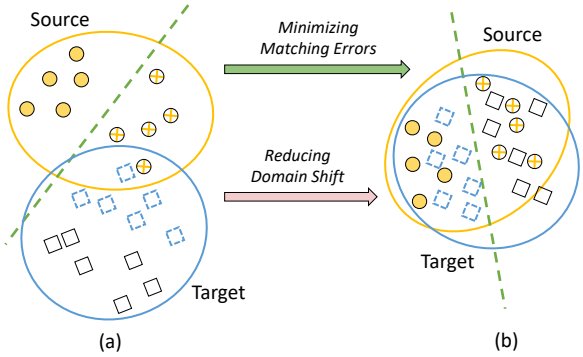


Figure 1: DA for ER. (a) A Matcher learned from *labeled* source cannot perform well on *unlabeled* target due to domain shift. (b) Domain adaptation learns better representations to reduce domain shift and minimize matching errors.

1 INTRODUCTION

Entity resolution (ER) determines whether two data instances refer to the same real-world entity. With the many decades of efforts of approaching ER from the model-centric point of view, there has been a considerable amount of literature, ranging from rule-based methods (*e.g.,* disjunctive normal form [59] and general boolean formula [59]), ML-based methods (such as SVM [5] and random forests [20]), to DL-based methods (such as DeepMatcher [49], DeepER [21], and Ditto [42]). The state-of-the-art (SOTA) results are, not surprisingly, achieved by DL-based solutions.

However, DL-based ER methods typically need a large amount of labeled training data. For example, even by piggybacking pre-trained language models such as Ditto [42], thousands of labels are still needed to achieve a satisfactory accuracy. In fact, the main pain point for ER practitioners is that they would need substantial labeling efforts for creating enough training data.

Fortunately, the big data era makes a lot of labeled ER datasets available in the same or relevant domains, either from public benchmarks (*e.g.,* WDC [52] and DBLP-Scholar [49]) or within enterprises. Hence, a natural question is: can we reuse these labeled *source* ER datasets for a new *target* ER dataset? An affirmative answer to the above question has the potential to dramatically reduce the expensive human effort for data labeling.

Domain adaptation. The key challenge of reusing labeled source data is that there may be distribution change or *domain shift* between the source and the target, which would degrade the performance. Figure 1 (a) shows an example of a labeled source dataset (circles) and an unlabeled target dataset (squares). As the source and target datasets may not be from the same domain, they do not

id	title	category	brand	price		id	title	category	brand	price
a_1^S	balt wheasel ...	stationery ...	balt	239.88	$(a_1^S, b_1^S, 1)$	b_1^S	balt inc. ...	laminating ...	mayline	134.45
a_2^S	kodak esp ...	printers	NULL	58.0	$(a_2^S, b_2^S, 0)$	b_2^S	kodak esp 7 ...	kodak	NULL	149.29
a_3^S	hp q3675a ...	printers	hp	194.84	$(a_3^S, b_3^S, 1)$	b_3^S	hewlett ...	cleaning repair	hp	NULL

(a) Labeled Source Dataset

id	name	description	price		id	name	description	price
a_1^T	samsung 52 ' series 7 black flat ...	samsung 52 ' series 7 black flat panel lcd ...	NULL	$(a_1^T, b_1^T, ?)$	b_1^T	samsung ln52a750 ...	dynamic contrast ratio 120hz 6ms respons ...	2148.99
a_2^T	sony 46 ' bravia ...	bravia z series ...	NULL	$(a_2^T, b_2^T, ?)$	b_2^T	sony bravia ...	ntsc 16:9 1366 x 768 ...	597.72
a_3^T	linksys wirelessn ...	security router ...	NULL	$(a_3^T, b_3^T, ?)$	b_3^T	linksys wirelessg ...	54mbps	NULL

(b) Unlabeled Target Dataset

Figure 2: A running example of DA for ER with a labeled source dataset $\mathcal{D}^S$ and an unlabeled target dataset $\mathcal{D}^T$.

follow the same distribution. As a result, an ER model (the green line) trained from the source cannot correctly predict the target. To address the challenge, domain adaptation (DA) is extensively studied to utilize labeled data in one or more relevant source domains for a new dataset in a target domain [25, 45, 64, 69]. Intuitively, DA is to learn from data instances what is the best way of aligning distributions of the source and the target data, such that the models trained on the labeled source can be used (or adapted) to the unlabeled target. As illustrate in Figure 1 (b), the advantage of DA is its ability to learn more *domain-invariant* representations that reduce the domain shift between source and target, and to improve performance of the ER model, *e.g.,* the green line can correctly classify data instances in both source and target datasets.

However, despite some very recent attempts [35], as far as we know, the adoption of DA in ER is not systematically studied under the same framework, and thus it is hard for practitioners to understand DA's benefits and limitations for ER. To bridge this gap, this paper introduces a general framework, called **DADER** (**D**omain **A**daptation for **D**eep **E**ntity **R**esolution) that unifies a wide range of choices of DA solutions [73, 74]. Specifically, the framework consists of three main modules. (1) *Feature Extractor* converts entity pairs to high-dimensional vectors (*a.k.a.* features). (2) *Matcher* is a binary classifier that takes the features of entity pairs as input, and predicts whether they match or not. (3) *Feature Aligner* is the key module for domain adaptation, which is designed to alleviate the effect of domain shift. To achieve this, Feature Aligner adjusts Feature Extractor to align distributions of source and target ER datasets, which then reduces domain shift between source and target. Moreover, it updates Matcher accordingly to minimize the matching errors in the adjusted feature space.

Design space exploration. Based on our framework, we systematically categorize and study the most representative methods in DA for ER, and focus on investigating two key questions.

First, DA is a broad topic in machine learning (*e.g.,* computer vision and natural language processing), and there is a large set of design choices for domain adaptation. Thus, it is necessary to ask a question that which DA design choices would help ER. To answer the question, we have extensively reviewed existing DA studies, and then focus on the most popular and fruitful directions that learn domain-invariant and discriminative features. Based on this, we provide a categorization for each module in **DADER** and define a *design space*, by summarizing representative DA techniques. Specifically, Feature Extractor is typically implemented by recurrent neural networks [38] and pre-trained language models [19, 44, 55]. Matcher often adopts a deep neural networks as a binary classifier. Feature Aligner is implemented by three categories of solutions: (1) discrepancy-based, (2) adversarial-based, and (3) reconstruction-based. As the concrete choices of Feature Extractor and Matcher have been well studied, our focus is to identify methods for Feature Aligner, for which we develop six representative methods that cover a wide range of SOTA DA techniques.

The second question is whether DA is useful for ER to utilize labeled data in relevant domains. To answer this, this paper considers two settings: (1) the unsupervised DA setting without any target labels, and (2) the semi-supervised DA setting with a few target labels. Moreover, we also compare **DADER** with SOTA DL solutions for ER, such as DeepMatcher [49] and Ditto [42]. Based on the comparison, we provide comprehensive analysis on the benefits and limitations of DA for ER.

Contributions: (1) As far as we know, we are the first to formally define the problem of **DA for deep ER** (Section 3) and conduct so far the most comprehensive study for applying DA to ER.

(2) We introduce a **DADER** framework that supports DA for ER, which consists of three modules, namely Feature Extractor, Matcher and Feature Aligner. We systematically explore the design space of DA for ER by categorizing each individual module in the framework (Section 4). In particular, we develop **six representative methods for Feature Aligner** (Section 5).

(3) We conduct a thorough evaluation to explore the design space and compare the developed methods (Section 6). The source code and data have been made available at Github[1]. We find that DA is very promising for ER, as it reduces domain shift between source and target. We point out some open problems of DA for ER and identify research directions (Section 8).

2 DEEP ENTITY RESOLUTION

We formally define entity resolution and present a framework of using deep learning for entity resolution (or *Deep ER* for short).

Entity resolution. Let A and B be two relational tables with multiple attributes. Each tuple $a \in A$ (or $b \in B$) is also referred to as an *entity* consisting of a set of attribute-value pairs $\{(\texttt{attr}_i, \texttt{val}_i)\}_{1 \le i \le k}$, where $\texttt{attr}_i$ and $\texttt{val}_i$ denote the i-th attribute name and value respectively. The problem of *entity resolution* (ER) is to find all the

[1] https://github.com/ruc-datalab/DADER

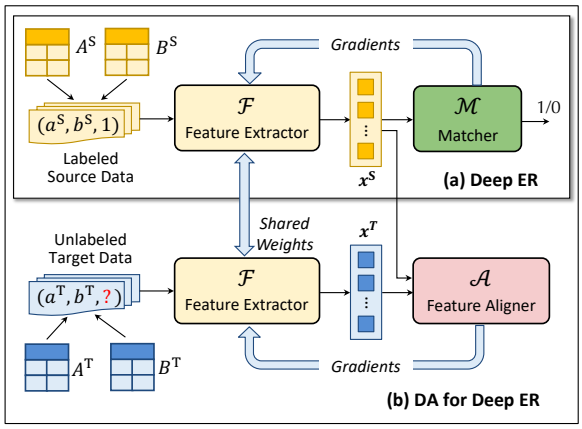


Figure 3: Domain adaptation for deep entity resolution. (a) A general framework for deep entity resolution. (b) Our DADER framework of domain adaption for deep entity resolution.

entity pairs $(a, b) \in A \times B$ that refer to the same real-world objects. An entity pair is said to be *matching* (resp. *non-matching*) if they refer to the same (resp. different) real-world objects.

A typical ER pipeline consists of two steps, *blocking* and *matching*. The blocking step generates a set of candidate pairs with high recall, *i.e.,* pruning the entity pairs which are unlikely to match (see [67] about DL for ER blocking). The matching step takes the candidate set generated from the blocking step as input and determines which candidate pairs are matchings or non-matchings. Our focus is on the domain adaption for the ER matching step.

Training data for ER. We denote a *labeled training set* as $(\mathcal{D}, \mathcal{Y})$, where $\mathcal{D} \subset A \times B$ is a set of entity pairs and $\mathcal{Y}$ is a label set. Each entity pair $(a, b) \in \mathcal{D}$ is associated with a label $y \in \mathcal{Y}$ that denotes whether the pair of entities a and b is matching (*i.e.,* $y = 1$) or non-matching (*i.e.,* $y = 0$). Figure 2 (a) shows an example training set. Each pair consists of two entities from different tables, *e.g.,* (a_1^S, b_1^S), and is associated with a $1/0$ label.

Deep entity resolution. Existing Deep ER solutions [21, 42, 49] typically utilize a framework that consists of a *Feature Extractor* and a *Matcher*, as shown in Figure 3 (a). Specifically, given an entity pair (a, b), a Feature Extractor $\mathcal{F}(a, b) : A \times B \rightarrow \mathbb{R}^d$, converts this pair into d-dimensional vector-based representation (*a.k.a.* features), denoted by $\mathbf{x}$, *i.e.,* $\mathbf{x} = \mathcal{F}(a, b)$. Then, features $\mathbf{x}$ will be fed into an *ER Matcher* $\mathcal{M}$, which is a DL-based binary classification model. The ER Matcher $\mathcal{M}$ takes *features* $\mathbf{x}$ as input, and predicts a probability $\hat{y}$ of matching,

$$\hat{y} = \mathcal{M}(\mathbf{x}) = \mathcal{M}(\mathcal{F}(a, b)). \quad (1)$$

Given a training set $(\mathcal{D}, \mathcal{Y})$, by iteratively applying minibatch stochastic gradient descent, parameters of both $\mathcal{F}$ and $\mathcal{M}$ are optimized, and thus they are improved to distinguish matching entity pairs from the non-matchings.

Example 1. *Consider the ER dataset in Figure 2 (a). Suppose that we use the pre-trained language model Bert [19] to implement Feature Extractor $\mathcal{F}$, like Ditto [42]. Given an entity a, $\mathcal{F}$ first serializes all attribute-value pairs $\{(\texttt{attr}_i, \texttt{val}_i)\}_{1 \le i \le k}$ of a into a token sequence* (i.e., *text) by applying the following function,*

$$\mathcal{S}(a) = \texttt{[ATT]}\ \texttt{attr}_1\ \texttt{[VAL]}\ \texttt{val}_1 \ldots \texttt{[ATT]}\ \texttt{attr}_k\ \texttt{[VAL]}\ \texttt{val}_k,$$

where [ATT] *and* [VAL] *are two special tokens for starting attributes and values respectively. For example, we serialize entity a_1^S into a token sequence,* i.e., $\mathcal{S}(a_1^S)$ = [ATT] `title` [VAL] `balt...` [ATT] `price` [VAL] `239.88` .

Then, $\mathcal{F}$ converts (a^S, b^S) into token sequence $\mathcal{S}(a^S, b^S)$ = [CLS] $\mathcal{S}(a^S)$ [SEP] $\mathcal{S}(b^S)$ [SEP]*, where* [SEP] *is a special token separating the two entities and* [CLS] *is a special token in Bert to encode the entire sequence. Finally, we feed $\mathcal{S}(a^S, b^S)$ into Bert and obtain a vector-based representation* $\mathbf{x}$ (e.g., *the embedding of* [CLS]*). After that, we can use a fully connected layer to implement Matcher $\mathcal{M}$, which then produces matching probability $\hat{y}$ from* $\mathbf{x}$*. Thus, given all the labeled pairs $\{(a^S, b^S, y^S)\}$ in Figure 2 (a), we can train both $\mathcal{F}$ and $\mathcal{M}$ by minimizing a loss function over $\{(y^S, \hat{y})\}$.*

3 DOMAIN ADAPTATION FOR DEEP ER

Next we describe domain adaptation for deep ER. We consider a *labeled* source ER dataset $(\mathcal{D}^S, \mathcal{Y}^S) = \{(a^S, b^S, y^S)\}$ and an *unlabeled* target dataset $\mathcal{D}^T = \{(a^T, b^T)\}$, and aim to find the best Feature Extractor and Matcher for producing accurate matching/non-matching results on target $\mathcal{D}^T$. A crude method is to directly use Feature Extractor $\mathcal{F}$ and Matcher $\mathcal{M}$ trained using $\mathcal{D}^S$ to predict $\mathcal{D}^T$. However, because source and target data may not come from the same domain, the features extracted by $\mathcal{F}$ for the source and the target may not follow the same data distribution, leading to a *domain shift* problem. Consequently, $\mathcal{M}$ trained using the source data cannot correctly predict the target data. To address this obstacle, we study the problem of *domain adaptation for ER*, and introduce a framework **DADER** that unifies the representative methods.

High-level idea of DA for ER. Figure 3 (b) shows the **DADER** framework. The high-level idea is to learn from data instances what is the best way of generating and aligning the features for the source and the target entity pairs, such that the Matcher trained on the labeled source can be used (or adapted) to the unlabeled target. To this end, **DADER** introduces a *Feature Aligner* $\mathcal{A}$, which guides Feature Extractor $\mathcal{F}$ to generate *domain-invariant* and *discriminative* features and updates Matcher $\mathcal{M}$ accordingly. Formally, we define the alignment loss and the matching loss as follows.

(1) Domain-invariant. Intuitively, we would like the distributions of source features $\mathbf{x}^S$ and target features $\mathbf{x}^T$ to be as close as possible. To this end, the Feature Aligner $\mathcal{A}(\mathbf{x}^S, \mathbf{x}^T) : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$, is utilized to produce an *alignment loss* $\mathcal{L}_A$. For example, a simple method for $\mathcal{L}_A$ is to define a distance between the means of source and target distributions. A more comprehensive design exploration for fulfilling $\mathcal{L}_A$ will be discussed in Section 4.

(2) Discriminative. We also consider a matching loss $\mathcal{L}_M$ that measures the difference between predicted and ground-truth results on the source. Recall that, given an entity pair (a^S, b^S) in the source dataset, our Matcher $\mathcal{M}$ makes a prediction as $\hat{y} = \mathcal{M}(\mathcal{F}(a^S, b^S))$. Thus, matching loss $\mathcal{L}_M$ is defined as $\mathcal{L}_M = \texttt{loss}(\{(\hat{y}, y^S)\})$, where `loss` is a function, such as cross entropy.

Now, we are ready to present the goal of DA for ER as finding the best Feature Extractor $\mathcal{F}^*$ and Matcher $\mathcal{M}^*$ that minimize both alignment loss $\mathcal{L}_A$ and matching loss $\mathcal{L}_M$, *i.e.,*

$$\mathcal{F}^*, \mathcal{M}^* = \arg\min_{\mathcal{F}, \mathcal{M}} \texttt{aggregate}(\mathcal{L}_A, \mathcal{L}_M). \quad (2)$$

Table 1: The design space of DA solutions for ER in this paper (MMD: Maximum Mean Discrepancy. K-order: K-order Statistics. GRL: Gradient Reversal Layer. InvGAN: Inverted Labels Generative Adversarial Network. KD: Knowledge Distillation. ED: Encoder-Decoder).

Modules	Categorization	
Feature Extractor ($\mathcal{F}$)	(I) Recurrent neural network (RNN) (II) Pre-trained language models (LMs)	
Matcher ($\mathcal{M}$)	Multi-layer Perceptron (MLP)	
Feature Aligner ($\mathcal{A}$)	(1) Discrepancy-based	(a) MMD (b) K-order
	(2) Adversarial-based	(c) GRL (d) InvGAN (e) InvGAN+KD
	(3) Reconstruction-based	(f) ED

Here, `aggregate` is a function that combines these two losses, which is determined by various feature alignment strategies. To achieve the objective in Equation (2), we leverage back-propagation to update parameters of $\mathcal{F}$ and $\mathcal{M}$, as shown in Figure 3. In such a way, $\mathcal{F}$ could be improved to produce domain-invariant features across source and target, while $\mathcal{M}$ can be updated accordingly to improve ER matching on both source and target datasets.

After that, both $\mathcal{M}$ and $\mathcal{F}$ are ready for use on the target dataset. Specifically, given each target entity pair $(a^\mathsf{T}, b^\mathsf{T})$, they can work together to make a prediction as $\mathcal{M}(\mathcal{F}(a^\mathsf{T}, b^\mathsf{T}))$.

Example 2. *Consider our example in Figure 2. Suppose that $\mathcal{F}$ maps entity pairs in both source (*i.e., *circles) and target (*i.e., *squares) into a 2-dimensional space, as shown in Figure 1 (a). Then, we can see an example of domain shift between source and target datasets. This may be because $\mathcal{F}$ pays much attention to the specific attributes in the source that are not in the target,* e.g., `category` *and* `brand`. *As a result, Matcher $\mathcal{M}$ (*i.e., *the dotted line) is not directly suitable for $\mathcal{D}^\mathsf{T}$ and produces non-match results for all the target entity pairs. To solve this problem, Feature Aligner $\mathcal{A}$ is introduced to achieve the following two objectives: (1) it adjusts $\mathcal{F}$ to align distributions of source and target entity pairs, e.g., making full use of the shared attributes, and (2) it updates $\mathcal{M}$ accordingly to minimize the matching errors in the newly generated feature space. Based on this, $\mathcal{A}$ makes the updated $\mathcal{F}$ and $\mathcal{M}$ to produce more accurate ER results, as shown in Figure 1 (b). For better illustration, we also provide several real examples on our experimental datasets in Section 6.2.1.*

Remarks. Note that domain adaptation is a broad topic in machine learning, and gains great success in both CV and NLP (cf. recent surveys [73, 74]). In our ER scenario, we adopt the most popular and fruitful family of domain adaptation techniques, *i.e.,* learning domain-invariant and discriminative features [25, 45, 69]. However, there is a larger set of choices of domain adaptation, and many DA techniques are not covered by this paper, *e.g.,* generating pseudo labels [26], reweighting source samples [8, 16], etc. Please refer to Section 7 for a more comprehensive discussion.

4 A DESIGN SPACE OF DA FOR DEEP ER

Entity resolution has its own characteristics that make the application of DA for ER challenging. First, each data instance in ER, *i.e.,* an entity pair, is complicated, as it represents two entities from relational tables with multiple attributes. Thus, it is non-trivial for Feature Extractor $\mathcal{F}$ to convert each entity pair to a vector-based representation. Second, it remains unexplored which DA methods are adequate for the ER scenario.

To address the challenges, we propose a categorization of design choices for each module in **DADER**, which forms a design space as shown in Table 1. We note that **DADER** is extensible, *i.e.,* it is possible to incorporate new modules, new categories, or new methods or variants of existing methods. Moreover, it is possible to define the search space from a different angle; that is, we contend that our proposal is rational, but may not be unique.

4.1 Design of Feature Aligner

Feature Aligner $\mathcal{A}$ is the key module to reduce the domain shift between source and target. Following the widely-recognized categorization of DA for CV and NLP [73, 74], we consider discrepancy-based, adversarial-based and reconstruction-based methods, as summarized in Table 1. The difference among these methods is how they define the alignment loss $\mathcal{L}_A$ that is defined in Section 3.

(1) Discrepancy-based methods measure alignment loss $\mathcal{L}_A$ by computing *distribution discrepancy* between source and target. Intuitively, considering our example in Figure 1, discrepancy-based methods aim at making the circles (source) and the squares (target) as close as possible. The key technical issue is how to define the distribution discrepancy. Existing works have proposed some statistical metrics, among which Maximum Mean Discrepancy (MMD) [29, 45] and K-orders [61, 62] are the most effective. Thus, we realize these two techniques as representative methods for discrepancy-based feature alignment.

(2) Adversarial-based methods measure alignment loss $\mathcal{L}_A$ by introducing a *domain classifier* and an adversarial training process inspired by generative adversarial networks (GAN) [28]. Intuitively, the domain classifier is trained to distinguish features from source or target, while the Feature Extractor $\mathcal{F}$ tries to generate the features that can confuse this domain classifier. We implement three representative adversarial-based methods, namely gradient reversal layer (GRL) [25], inverted labels GAN (InvGAN) [53, 69] and InvGAN+KD (an improved version of InvGAN).

(3) Reconstruction-based methods fulfill alignment loss $\mathcal{L}_A$ by introducing an auxiliary *unsupervised reconstruction task*. Specifically, Feature Aligner $\mathcal{A}$ is used as the decoder to reconstruct the input of feature extractor $\mathcal{F}$, to ensure that features extracted by $\mathcal{F}$ contain useful and shared information across source and target. This technique has achieved superior performance in CV [27, 73]. We implement the Encoder-Decoder (ED) networks [39], which realize the reconstruction task in NLP as the representative method.

We will describe design details of the above discrepancy-based, adversarial-based and reconstruction-based methods in Section 5.

4.2 Design of Feature Extractor and Matcher

Feature Extractor $\mathcal{F}$. Recall that Feature Extractor $\mathbf{x} = \mathcal{F}(a, b)$ aims at representation learning, *i.e.,* converting a pair of entities a and b into a vector-based representation $\mathbf{x}$. Existing studies for deep ER utilize the following two common and effective solutions.

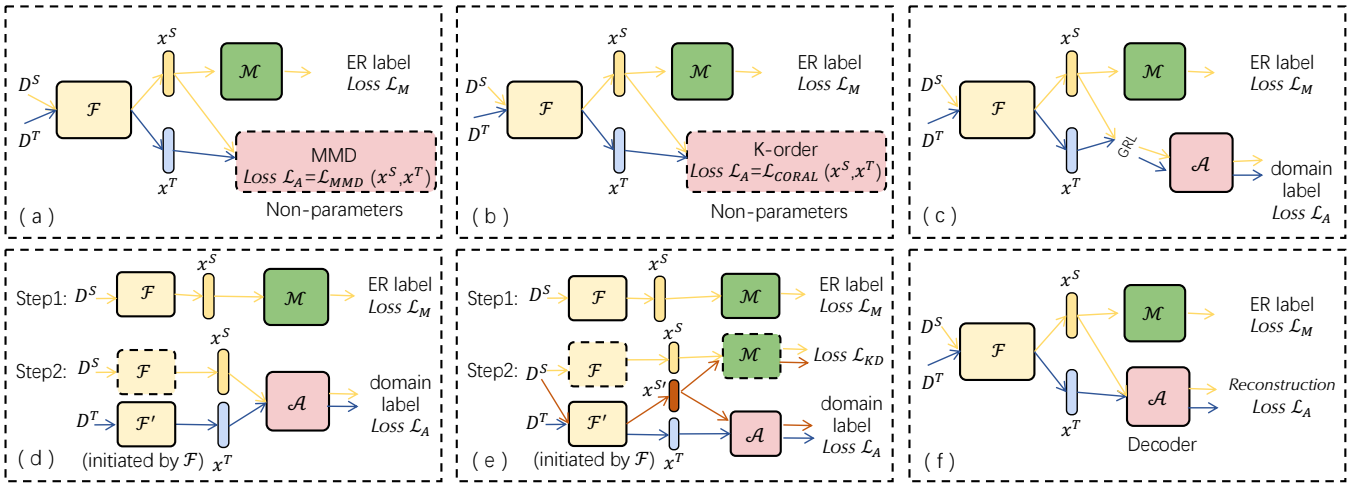


Figure 4: Architectures of six representative Feature Aligner methods. (a) Maximum Mean Discrepancy (MMD) and (b) K-order are *discrepancy-based*; (c) Gradient Reversal Layer (GRL), (d) Inverted Labels GAN (InvGAN) and (e) InvGAN + Knowledge Distillation (KD) are *adversarial-based*; and (f) Encoder-Decoder (ED) is *reconstruction-based*.

(1) Recurrent neural network (RNN) is a well-adopted solution for encoding variable-length text into features, and achieves good performance in NLP tasks [57, 77] and ER [21, 35, 49]. Now we specifically describe the process of RNN feature extraction. Given an entity pair (a, b) with k aligned attributes, the RNN network will be used to compute an embedding for each attribute value, resulting in k embeddings for each entity, *i.e.,* $\{\mathbf{e}_1^a, \ldots, \mathbf{e}_k^a\}$ for a and $\{\mathbf{e}_1^b, \ldots, \mathbf{e}_k^b\}$ for b, which are known as *attribute embeddings*. Afterwards, it computes upon the above two sets of embeddings, one *entity similarity embedding* to be used by the Matcher. To do this, there are two ways: (1) It computes a similarity vector $\mathbf{s}_i$ for each pair of attributes $\mathbf{e}_i^a$ and $\mathbf{e}_i^b$, for $i \in [1, k]$ and then aggregates these k similarity vectors into one entity pair similarity embedding $\mathbf{s}$. (2) It aggregates the k attribute embeddings for each entity into one entity embedding, *e.g.,* aggregating $\{\mathbf{e}_1^a, \ldots, \mathbf{e}_k^a\}$ (resp. $\{\mathbf{e}_1^b, \ldots, \mathbf{e}_k^b\}$) and resulting in $\mathbf{e}^a$ (resp. $\mathbf{e}^b$). Then, $\mathbf{e}^a$ and $\mathbf{e}^b$ will be used to compute the entity pair similarity embedding $\mathbf{s}$. Note that the entity pair similarity embedding $\mathbf{s}$ is the feature $\mathbf{x}$ we need. Please refer to the existing ER studies [21, 35, 49] for more details.

(2) Pre-trained language models (LMs) have achieved superior performance for representing entity pairs in ER [42, 65]. Representative LMs include Bert [19], RoBerta [44], DistilBert [55], etc. We take Bert as an example to illustrate how it encodes an entity pair (a, b) into a vector-based representation, while the other LMs are very similar. Bert provides a simple text feature extraction model. Please refer to Example 1 that illustrates how Bert works for ER.

Matcher $\mathcal{M}$. Matcher $\mathcal{M}$ is a binary classifier, which takes a feature $\mathbf{x}$ as input and classifies it as 1 (matching) or 0 (non-matching). In this work, we adopt MLP, the most common choice used in existing DL-based ER (*e.g.,* DeepMatcher [49], DeepER [21] and Ditto [42]). Note that we do not consider other choices for $\mathcal{M}$, because the design of Matcher is not the focus of domain adaptation for ER.

5 DESIGN CHOICES OF FEATURE ALIGNER

We describe the six representative methods for Feature Aligner. Figure 4 shows the architectures of six representative methods, where arrows with different colors represent the data flows of source/target datasets. During training, loss is calculated and the corresponding network parameters are updated. Note that only the parameters in the solid line boxes need to be updated, and the dotted line boxes indicate that their parameters do not need to be updated or they have no parameters. After training, the obtained Matcher $\mathcal{M}$ and Feature Extractor $\mathcal{F}$ are used for target dataset $\mathcal{D}^T$ for prediction. Briefly, we use x^S and x^T to represent entity pairs (a^S, b^S) and (a^T, b^T), respectively. The corresponding feature spaces generated by $\mathcal{F}$ are denoted as p_S and p_T, respectively.

5.1 Discrepancy-based Methods

Figures 4 (a) and (b) correspond to the discrepancy-based methods, MMD and K-order, respectively. The high-level idea is to use statistical metrics to minimize the domain distribution discrepancy between source and target. Thus, Feature Aligner $\mathcal{A}$ is a fixed function to calculate the discrepancy value, without parameters. During training, $\mathcal{D}^S$ and $\mathcal{D}^T$ are fed into $\mathcal{F}$ for generating features $\{\mathbf{x}^S\}$ and $\{\mathbf{x}^T\}$, and their feature spaces are p_S and p_T. Then, $\mathcal{A}$ computes a distribution discrepancy (denoted as $\mathcal{L}_A$) between p_S and p_T. Meanwhile, Matcher $\mathcal{M}$ gives ER label prediction and computes the matching loss $\mathcal{L}_M$ over labeled data $\mathcal{D}^S$. The optimization objective is to simultaneously reduce $\mathcal{L}_A$ and $\mathcal{L}_M$, with the goal of learning domain-invariant and discriminative features, *i.e.,*

$$\min_{\mathcal{F},\mathcal{M},\mathcal{A}} V(\mathcal{F}, \mathcal{M}, \mathcal{A}) = \mathcal{L}_M(\mathcal{F}, \mathcal{M}) + \beta \mathcal{L}_A(\mathcal{F}, \mathcal{A}), \tag{3}$$

$$\mathcal{L}_M = E_{(x^S, y^S)\sim(\mathcal{D}^S, \mathcal{Y}^S)}[\mathcal{L}_{CE}(\mathcal{M}(\mathcal{F}(x^S)), y^S)]. \tag{4}$$

where $\mathcal{L}_{CE}$ is the cross-entropy loss that is commonly used in DL, and β is the hyper-parameter that controls the trade-off between matching and domain confusion. $E[\cdot]$ denotes the expectation.

Based on Equation (3), we introduce Algorithm 1 as a template to unify the discrepancy-based, GRL-based and reconstruction-based methods (the latter two will be discussed later).

***Algorithm 1*:** Line 1 first initializes the Feature Extractor, the Matcher and the Feature Aligner. For each iteration (lines 2–13), lines 3–4 first sample one mini-batch from the labeled source and one mini-batch from the unlabeled target. Lines 5–6 compute the loss of the Matcher according to Equation (4) and use

Algorithm 1 Discrepancy/GRL/Reconstruction-based Methods

Input: $(\mathcal{D}^{\mathsf{S}}, \mathcal{Y}^{\mathsf{S}})$: labeled source data; $\mathcal{D}^{\mathsf{T}}$: unlabeled target data; β: the weight of alignment loss; μ: learning rate
Output: $\mathcal{F}$, $\mathcal{M}$.

1: Initialize $\mathcal{F}$, $\mathcal{M}$ and $\mathcal{A}$;
2: **for** pre-defined number of iterations **do**
3: Sample one minibatch $\mathsf{BT}^{\mathsf{S}} = \{(x^{\mathsf{S}}, y^{\mathsf{S}})\}$ from $(\mathcal{D}^{\mathsf{S}}, \mathcal{Y}^{\mathsf{S}})$;
4: Sample one minibatch $\mathsf{BT}^{\mathsf{T}} = \{x^{\mathsf{T}}\}$ from $\mathcal{D}^{\mathsf{T}}$;
5: Compute $\mathcal{L}_M$ according to Equation (4);
6: $\theta_{\mathcal{M}} \leftarrow \theta_{\mathcal{M}} - \mu \frac{\partial \mathcal{L}_M}{\partial \theta_{\mathcal{M}}}$;
7: **if** Discrepancy/Reconstruction-based methods **then**
8: NoAdvAdapt($\mathcal{F}$, $\mathcal{A}$, BT^{S}, BT^{T}, $\mathcal{L}_M$);
9: **end if**
10: **if** GRL-based methods **then**
11: GRLAdapt($\mathcal{F}$, $\mathcal{A}$, BT^{S}, BT^{T}, $\mathcal{L}_M$);
12: **end if**
13: **end for**

Procedure 1 NoAdvAdapt ($\mathcal{F}$, $\mathcal{A}$, BT^{S}, BT^{T}, $\mathcal{L}_M$)

1: Generate features by $\mathcal{F}$ and compute $\mathcal{L}_A$ using Equation (5), (6) for MMD and K-order, or compute $\mathcal{L}_A$ using Equation (15) for reconstruction-based methods;
2: $\theta_{\mathcal{A}} \leftarrow \theta_{\mathcal{A}} - \mu\beta \frac{\partial \mathcal{L}_A}{\partial \theta_{\mathcal{A}}}$; /* update only if $\mathcal{A}$ is a neural network*/
3: $\theta_{\mathcal{F}} \leftarrow \theta_{\mathcal{F}} - \mu(\frac{\partial \mathcal{L}_M}{\partial \theta_{\mathcal{F}}} + \beta \frac{\partial \mathcal{L}_A}{\partial \theta_{\mathcal{F}}})$;

back-propagation to tune the Matcher. Lines 7–9 are for discrepancy and reconstruction-based methods, which invoke Procedure 1 NoAdvAdapt. Lines 10–12 are for GRL-based methods that invoke Procedure 2 GRLAdapt (which will be discussed later).

In the procedure NoAdvAdapt, line 1 computes alignment loss $\mathcal{L}_A$, depending on different methods. Then, line 2 uses back-propagation to tune Feature Aligner only if it is a neural network; otherwise, this line will not be executed as the Feature Aligner is a fixed function. Finally, line 3 uses back-propagation to tune Feature Extractor by considering a linear combination of $\mathcal{L}_M$ and $\mathcal{L}_A$.

Next we describe the characteristics of MMD and K-order, and explain how they implement the calculation of $\mathcal{L}_A$.

(a) MMD. MMD is an effective metric to measure distribution discrepancy by a kernel two-sample test [29]. If and only if the two distributions are the same, the value of MMD is 0. Because of the characteristic of MMD, the MMD metric can be computed between the features of two domains (denoted by $\mathcal{L}_{\mathtt{MMD}}$) to reduce the distribution mismatch in the latent space, which is defined as:

$$\mathcal{L}_{\mathtt{MMD}} = \sup_{\|\phi\|_H \le 1} \|E_{\mathbf{x}^{\mathsf{S}} \sim p_{\mathsf{S}}}[\phi(\mathbf{x}^{\mathsf{S}})] - E_{\mathbf{x}^{\mathsf{T}} \sim p_{\mathsf{T}}}[\phi(\mathbf{x}^{\mathsf{T}})]\|_H^2, \quad (5)$$

where ϕ represents the kernel function that maps $\mathbf{x}^{\mathsf{S}}(\mathbf{x}^{\mathsf{T}})$ to a reproducing kernel Hilbert space (RKHS). $\|\phi\|_H \le 1$ defines a set of functions in the unit ball of RKHS (H). In particular, when the distributions of p_{S} and p_{T} are same, $\mathcal{L}_{\mathtt{MMD}}$ is zero.

(b) K-order. K-order is another effective metric to reduce distribution mismatch of domains. CORAL [61] learns a linear transformation that aligns the second-order statistics (covariance) between domains. DeepCORAL [62] learns a nonlinear transformation based on CORAL, and achieves good performance. More specifically, the distance between the second-order statistics of the two domains (denoted as $\mathcal{L}_{\mathtt{CORAL}}$) is defined as:

$$\mathcal{L}_{\mathtt{CORAL}} = \frac{1}{4d^2}\|C_{\mathsf{S}} - C_{\mathsf{T}}\|_F^2, \quad (6)$$

where d is the dimention of $\mathbf{x}$, $C_{\mathsf{S}}(C_{\mathsf{T}})$ is the covariance matrices calculated by the features $\mathbf{x}^{\mathsf{S}}(\mathbf{x}^{\mathsf{T}})$, and $\|\cdot\|_F^2$ denotes the squared matrix Frobenius norm. More detailed equation derivation can be found in DeepCORAL [62].

As shown in Figures 4 (a, b), $\mathcal{L}_A$ can be directly computed by $\mathcal{L}_{\mathtt{MMD}}$ and $\mathcal{L}_{\mathtt{CORAL}}$. As these two Feature Aligner methods are non-parameterized, only $\mathcal{F}$ and $\mathcal{M}$ will be tuned during training.

5.2 Adversarial-based Methods

Figures 4 (c, d, e) correspond to the three representative methods of adversarial-based: GRL, InvGAN and InvGAN+KD. Because they need to use a domain classifier to distinguish features from source or target, Feature Aligner $\mathcal{A}$ is a binary classifier implemented by fully-connected layers. During training, the optimization objective of $\mathcal{A}$ is to minimize the domain classification loss (which is denoted by $\mathcal{L}_A$ now), while $\mathcal{F}$ is generating the indistinguishable features that confuse $\mathcal{A}$. There are different training modes to achieve this goal, among which GRL-based and GAN-based are the most popular. The adversarial objective function is:

$$\min_{\mathcal{F},\mathcal{M}} \max_{\mathcal{A}} V(\mathcal{F}, \mathcal{M}, \mathcal{A}) = \mathcal{L}_M(\mathcal{F}, \mathcal{M}) + \beta \mathcal{L}_A(\mathcal{F}, \mathcal{A}), \quad (7)$$

$$\mathcal{L}_A = E_{x^{\mathsf{S}} \sim \mathcal{D}^{\mathsf{S}}} \log \mathcal{A}(\mathcal{F}(x^{\mathsf{S}})) + E_{x^{\mathsf{T}} \sim \mathcal{D}^{\mathsf{T}}} \log(1 - \mathcal{A}(\mathcal{F}(x^{\mathsf{T}}))), \quad (8)$$

where β is the hyper-parameter that controls the trade-off between matching and domain classification, and $\mathcal{L}_A$ is the adversarial loss.

(c) GRL. As shown in Figure 4 (c), this method adds a gradient reversal layer between $\mathcal{F}$ and $\mathcal{A}$, which is the key layer to realize the minimax adversarial training objective between $\mathcal{F}$ and $\mathcal{A}$. The gradient reversal layer has no parameters to update, which acts as an identity transformation in forward-propagation. $\mathcal{A}$ gives domain prediction for $\mathbf{x}^{\mathsf{S}}$ and $\mathbf{x}^{\mathsf{T}}$ and computes domain classification loss $\mathcal{L}_A$. During back-propagation, the gradient from $\mathcal{A}$ multiplies a certain negative constant in the gradient reversal layer. Meanwhile, the labeled $\mathbf{x}^{\mathsf{S}}$ is inputted into $\mathcal{M}$ and computes matching loss $\mathcal{L}_M$. Specifically, the loss function of GRL is :

$$\max_{\mathcal{F}} \min_{\mathcal{A}} \mathcal{L}_A^{(1)} = -E_{x^{\mathsf{S}} \sim \mathcal{D}^{\mathsf{S}}} \log(\mathcal{A}(\mathcal{F}(x^{\mathsf{S}}))) - E_{x^{\mathsf{T}} \sim \mathcal{D}^{\mathsf{T}}} \log(1 - \mathcal{A}(\mathcal{F}(x^{\mathsf{T}}))). \quad (9)$$

Algorithm 1 can also be applied to GRL where lines 10–12 are specially designed for GRL-based methods, *i.e.,* Procedure 2 GRLAdapt.

***Procedure 2*:** First, lines 1 computes alignment loss $\mathcal{L}_A$ according to Equation (9). Then, line 2 uses back-propagation to tune Feature Aligner. Finally, line 3 uses back-propagation to tune Feature Extractor, where the gradient for $\mathcal{F}$ from $\mathcal{A}$ has been reversed via multiplying a parameter $-\beta$.

(d) Inverted Labels GAN (InvGAN): To achieve the adversarial training objective, we adopt the original GAN-based architecture

Procedure 2 GRLAdapt ($\mathcal{F}$, $\mathcal{A}$, BT^{S}, BT^{T}, $\mathcal{L}_M$)

Compute $\mathcal{L}_A$ according to Equation (9);
$\theta_{\mathcal{A}} \leftarrow \theta_{\mathcal{A}} - \mu\beta\frac{\partial \mathcal{L}_A}{\partial \theta_{\mathcal{A}}}$;
$\theta_{\mathcal{F}} \leftarrow \theta_{\mathcal{F}} - \mu(\frac{\partial \mathcal{L}_M}{\partial \theta_{\mathcal{F}}} - \beta\frac{\partial \mathcal{L}_A}{\partial \theta_{\mathcal{F}}})$;

as one representative method, which uses *inverted labels* [28] to learn Feature Extractor. The architecture is proposed in ADDA [69] in CV and proved to be effective. The original GAN consists of two parts: Generator and Discriminator. The training process of GAN is to use the Generator to generate "fake data" to simulate the "real data", and the Discriminator is trained to distinguish them. The two parts are trained alternately. In our scenario, we regard $\mathbf{x}^{\mathsf{S}}$ as the "real data", Feature Extractor as the Generator that makes $\mathbf{x}^{\mathsf{T}}$ and $\mathbf{x}^{\mathsf{S}}$ as similar as possible (*i.e.,* these features are domain invariant), and regard Feature Aligner as Discriminator.

To implement the training process, as shown in Figure 4 (d), there are two training steps for ER problem. The first step is using labeled source data to train Feature Extractor $\mathcal{F}$ and Matcher $\mathcal{M}$. Hence, $\mathcal{M}$ can give accurate prediction on $\mathbf{x}^{\mathsf{S}}$. The second step is as follows. Since we need to use $\mathbf{x}^{\mathsf{S}}$ as our "real data", which is generated by the original $\mathcal{F}$, we clone a $\mathcal{F}'$ from $\mathcal{F}$. In the following training process, we train $\mathcal{F}'$ and keep $\mathcal{F}$ unchanged. We consider $\mathcal{F}'$ as the Generator, and $\mathcal{A}$ as the Discriminator. At first, x^{S} is inputted into $\mathcal{F}$ to generate the "real example" $\mathbf{x}^{\mathsf{S}}$, and x^{T} is inputted into $\mathcal{F}'$ to generate the "fake example" $\mathbf{x}^{\mathsf{T}}$. The goal is to train parameters of $\mathcal{F}'$ and $\mathcal{A}$ to generate $\mathbf{x}^{\mathsf{T}}$ and $\mathbf{x}^{\mathsf{S}}$ that are alike. Specifically, $\mathcal{A}$ is trained according to the following loss function:

$$\min_{\mathcal{A}} \mathcal{L}_A^{(2)} = -E_{x^{\mathsf{S}}\sim\mathcal{D}^{\mathsf{S}}}\log(\mathcal{A}(\mathcal{F}(x^{\mathsf{S}}))) - E_{x^{\mathsf{T}}\sim\mathcal{D}^{\mathsf{T}}}\log(1-\mathcal{A}(\mathcal{F}'(x^{\mathsf{T}}))). \tag{10}$$

$\mathcal{F}'$ is trained with inverted labels with the loss function as

$$\min_{\mathcal{F}'} \mathcal{L}_{F'}^{(1)} = -E_{x^{\mathsf{T}}\sim\mathcal{D}^{\mathsf{T}}}\log(\mathcal{A}(\mathcal{F}'(x^{\mathsf{T}}))). \tag{11}$$

Algorithm 2: We propose an algorithm template for InvGAN. Line 1 first initializes the Feature Extractor, the Matcher and the Feature Aligner. It then contains two steps. **Step 1** is to *only* train $\mathcal{F}$ and $\mathcal{M}$ using the labeled source, such that $\mathcal{M}$ can converge on the source feature $\mathbf{x}^{\mathsf{S}}$ (lines 2-7). Line 8 initializes $\mathcal{F}'$ with the parameters of $\mathcal{F}$. **Step 2** is for adversarial training, which adjusts $\mathcal{F}'$ to make $\mathbf{x}^{\mathsf{T}}$ and $\mathbf{x}^{\mathsf{S}}$ as similar as possible (lines 9-16).

One problem of InvGAN is that the $\mathbf{x}^{\mathsf{T}}$ may lose all discriminative information (*e.g.,* generating constants), which results in the performance degradation of $\mathcal{M}$. AAD [53] proposes to add one Kullback-Leibler divergence loss to avoid this problem.

(e) InvGAN+Knowledge Distillation (KD) [32]. To alleviate the above problem of possibly losing all discriminative information, we propose to use KD that learns a new "student" model from the "teacher" model and retains the classification ability of teacher model. This is applicable to our scenario because we want the performance of $\mathcal{F}'$ and $\mathcal{M}$ to be close to that of $\mathcal{F}$ and $\mathcal{M}$. The loss function of KD is:

$$\mathcal{L}_{\mathsf{KD}} = t^2 \times E_{x^{\mathsf{S}}\sim\mathcal{D}^{\mathsf{S}}} \sum -\mathtt{softmax}(\mathcal{M}(\mathcal{F}(x^{\mathsf{S}}))/t) \times \log(\mathtt{softmax}(\mathcal{M}(\mathcal{F}'(x^{\mathsf{S}}))/t)), \tag{12}$$

Algorithm 2 GAN-based Methods: InvGAN/InvGAN+KD

Input: $(\mathcal{D}^{\mathsf{S}}, \mathcal{Y}^{\mathsf{S}})$: labeled source data; $\mathcal{D}^{\mathsf{T}}$: unlabeled target data; β: the weight of alignment loss; μ: learning rate
Output: $\mathcal{F}'$, $\mathcal{M}$.

1: Initial $\mathcal{F}$, $\mathcal{M}$ and $\mathcal{A}$;
2: **for** pre-defined number of iterations **do**
3: Sample one minibatch $\mathsf{BT}^{\mathsf{S}} = \{(x^{\mathsf{S}}, y^{\mathsf{S}})\}$ from $(\mathcal{D}^{\mathsf{S}}, \mathcal{Y}^{\mathsf{S}})$;
4: Compute $\mathcal{L}_M$ according to Equation (4);
5: $\theta_{\mathcal{M}} \leftarrow \theta_{\mathcal{M}} - \mu\frac{\partial \mathcal{L}_M}{\partial \theta_{\mathcal{M}}}$;
6: $\theta_{\mathcal{F}} \leftarrow \theta_{\mathcal{F}} - \mu\frac{\partial \mathcal{L}_M}{\partial \theta_{\mathcal{F}}}$;
7: **end for**
8: $\mathcal{F}' \leftarrow \mathcal{F}$;
9: **for** pre-defined number of iterations **do**
10: Sample one minibatch $\mathsf{BT}^{\mathsf{S}} = \{(x^{\mathsf{S}}, y^{\mathsf{S}})\}$ from $(\mathcal{D}^{\mathsf{S}}, \mathcal{Y}^{\mathsf{S}})$;
11: Sample one minibatch $\mathsf{BT}^{\mathsf{T}} = \{x^{\mathsf{T}}\}$ from $\mathcal{D}^{\mathsf{T}}$;
12: Compute $\mathcal{L}_A$ according to Equation (10) or (13);
13: $\theta_{\mathcal{A}} \leftarrow \theta_{\mathcal{A}} - \mu\beta\frac{\partial \mathcal{L}_A}{\partial \theta_{\mathcal{A}}}$;
14: Compute $\mathcal{L}_{F'}$ according to Equation (11) or (14);
15: $\theta_{\mathcal{F}'} \leftarrow \theta_{\mathcal{F}'} - \mu\beta\frac{\partial \mathcal{L}_{F'}}{\partial \theta_{\mathcal{F}'}}$;
16: **end for**

where t is a hyper-parameter to determine how much the model *softens* [32] the distributions. $\mathcal{L}_{\mathsf{KD}}$ can be derived from the Kullback-Leibler divergence of the predicted distribution by $\mathcal{M}(\mathcal{F}(\cdot))$ and $\mathcal{M}(\mathcal{F}'(\cdot))$ since the $\mathcal{M}(\mathcal{F}(\cdot))$ is fixed during training.

In the adaptation step (*i.e.,* **Step 2**) in Figure 4 (e), $\mathcal{F}'$ is initialized by $\mathcal{F}$ at beginning, then x^{S} is inputted into $\mathcal{F}$ and $\mathcal{F}'$ to generate $\mathbf{x}^{\mathsf{S}}$ and $\mathbf{x}^{\mathsf{S}'}$, which are inputted into $\mathcal{M}$ to compute $\mathcal{L}_{\mathsf{KD}}$ according to Equation (12). The $\mathcal{L}_{\mathsf{KD}}$ can ensure that the features generated by $\mathcal{F}'$ can be distinguished by $\mathcal{M}$, *i.e.,* learning discriminative features, while the confrontation between $\mathcal{F}'$ and $\mathcal{A}$ can learn domain-invariant features. We use $\mathbf{x}^{\mathsf{S}'}$ and $\mathbf{x}^{\mathsf{T}}$ generated by $\mathcal{F}'$ to train $\mathcal{A}$, so the loss functions for $\mathcal{A}$ and $\mathcal{F}'$ are:

$$\min_{\mathcal{A}} \mathcal{L}_A^{(3)} = -E_{x^{\mathsf{S}}\sim\mathcal{D}^{\mathsf{S}}}\log(\mathcal{A}(\mathcal{F}'(x^{\mathsf{S}}))) - E_{x^{\mathsf{T}}\sim\mathcal{D}^{\mathsf{T}}}\log(1-\mathcal{A}(\mathcal{F}'(x^{\mathsf{T}}))), \tag{13}$$

$$\min_{\mathcal{F}'} \mathcal{L}_{F'}^{(2)} = -E_{x^{\mathsf{T}}\sim\mathcal{D}^{\mathsf{T}}}\log(\mathcal{A}(\mathcal{F}'(x^{\mathsf{T}}))) + \mathcal{L}_{\mathsf{KD}}. \tag{14}$$

5.3 Reconstruction-based Methods

As shown in Figure 4 (f), the Feature Aligner is realized as a Decoder, which reconstructs the input data $\mathcal{D}^{\mathsf{S}}$ and $\mathcal{D}^{\mathsf{T}}$. The reconstruction task ensures to learn a shared hidden representation space between domains. This is intuitive that the auxiliary reconstruction task can facilitate the learning of domain-invariant features – the reconstruction loss $\mathcal{L}_{\mathsf{REC}}$ can ensure the shared $\mathcal{F}$ to extract important and shared information from both domains. Meanwhile, the matching loss $\mathcal{L}_M$ can ensure that $\mathcal{M}$ works for the shared feature.

(f) Encoder-Decoder (ED). The Encoder-Decoder network is suitable for reconstruction-based methods, which takes $\mathcal{F}$ as the Encoder and $\mathcal{A}$ as the Decoder. The existing text Encoder-Decoder models include VAE [56], Bart [39], SDA [72], etc. Algorithm 1 provides a template for this. For each minibatch, x^{S} and x^{T} are inputted into $\mathcal{F}$ to generate features $\mathcal{F}(x^{\mathsf{S}})$ and $\mathcal{F}(x^{\mathsf{T}})$, and $\mathcal{F}(x^{\mathsf{S}})$ is inputted into $\mathcal{M}$ to get matching loss $\mathcal{L}_M$. Then, in Procedure 1,

$\mathcal{A}$ decodes these features to recover data x^{S} and x^{T}, which are then used to compute the reconstruction loss $\mathcal{L}_{\mathsf{REC}}$ according to Equation (15). Then the parameters can be updated.

$$\mathcal{L}_{\mathsf{REC}} = E_{x \sim \mathcal{D}^{\mathsf{S}} \cup \mathcal{D}^{\mathsf{T}}} [\mathcal{L}_{CE}(\mathcal{A}(\mathcal{F}(x)), x)] \tag{15}$$

Take Bart [39] as an example, the architecture consists of a bi-directional transformer Encoder and an auto-regressive transformer Decoder. The Encoder is used as Feature Extractor, which generates hidden representation for the original input text, and the Decoder inputs the hidden representation, then outputs the reconstructed text. The reconstruction loss $\mathcal{L}_{\mathsf{REC}}$ can be calculated between the generated and the original text, *i.e.,* the entity pairs. More details of the architecture can be referred to the original paper of Bart [39].

6 EVALUATION

We present the experimental setup in Section 6.1 and report the overall result of DA for ER in Section 6.2. Then, we explore the design space of **DADER** by evaluating Feature Aligner in Section 6.3 and Feature Extractor in Section 6.4. Next, we compare our framework with the state-of-the-art DL-based ER approaches in Section 6.5.

6.1 Experimental Setup

Datasets: We use the benchmark datasets from DeepMatcher [49] and Magellan [20], which cover a variety of domains, such as product, citation and restaurant. Each dataset contains entities from two relational tables with multiple attributes, and a set of labeled matching/non-matching entity pairs. Take the DBLP-Scholar (DS) dataset in Table 2 as an example: there are two tables extracted from DBLP and Scholar respectively, each of which has four aligned attributes (`Title`, `Authors`, `Venue`, `Year`). This dataset contains 28,707 entity pairs, where 5,347 entity pairs are labeled as matching and the remaining pairs are non-matching. Moreover, we also consider four WDC product datasets [52], which are also used in Ditto [42]. The WDC datasets are collected from e-commerce websites, which contain four categories: computers, cameras, watches, and shoes, where each category has 1100 labeled entity pairs. We also note that, since the original version of Zomato-Yelp dataset is simple and all the methods perform well on this dataset, following the existing work DeepMatcher [49], we utilized a dirty version of the Zomato-Yelp dataset for evaluation. Please refer to Table 2 for more details of the datasets. For ease of presentation, we use the notation $\mathcal{D}^{\mathsf{S}} \rightarrow \mathcal{D}^{\mathsf{T}}$, *e.g.,* Walmart-Amazon $\rightarrow$ Abt-Buy, to represent that $\mathcal{D}^{\mathsf{S}}$ is the source and $\mathcal{D}^{\mathsf{T}}$ is the target.

Evaluation method. We implement our **DADER** framework of DA for ER, as shown in Figure 3. To evaluate the performance of DA for ER, we select one dataset as labeled source $(\mathcal{D}^{\mathsf{S}}, \mathcal{Y}^{\mathsf{S}})$ and another dataset as unlabeled target $\mathcal{D}^{\mathsf{T}}$. We split target dataset $\mathcal{D}^{\mathsf{T}}$ into a validation set $\mathcal{D}^{\mathsf{T}}_{\mathtt{val}}$ and a test set $\mathcal{D}^{\mathsf{T}}_{\mathtt{test}}$ with the ratio of 1:9. Note that labels of test target dataset $\mathcal{D}^{\mathsf{T}}_{\mathtt{test}}$ are not used in **DADER** for domain adaptation. Instead, we use the labels as ground-truth for evaluating the performance. Based on this, we evaluate a method of DA for ER in the following way.

First, we train our framework based on $(\mathcal{D}^{\mathsf{S}}, \mathcal{Y}^{\mathsf{S}})$ and $\mathcal{D}^{\mathsf{T}}$. For fairly comparing the design solutions presented in Section 4, during the training process, we divide the training iterations into 40 epochs and evaluate the performance of DA model snapshot of each epoch on the validation set $\mathcal{D}^{\mathsf{T}}_{\mathtt{val}}$. We select the DA model snapshot with the best performance across all epochs, and use the corresponding optimized Feature Extractor $\mathcal{F}$ and Matcher $\mathcal{M}$. After that, we use the obtained Feature Extractor $\mathcal{F}$ and Matcher $\mathcal{M}$ to make prediction on the target dataset $\mathcal{D}^{\mathsf{T}}_{\mathtt{test}}$ to produce matching/non-matching results for evaluation.

Table 2: Real-world ER datasets for our evaluation: #Pairs, #Matches and #Attrs represent the numbers of entity pairs, matching pairs, and attributes respectively.

Datasets	Domain	#Pairs	#Matches	#Attrs
Walmart-Amazon (WA)	Product	10,242	962	5
Abt-Buy (AB)	Product	9,575	1,028	3
DBLP-Scholar (DS)	Citation	28,707	5,347	4
DBLP-ACM (DA)	Citation	12,363	2,220	4
Fodors-Zagats (FZ)	Restaurant	946	110	6
Zomato-Yelp (ZY)	Restaurant	894	214	3
iTunes-Amazon (IA)	Music	532	132	8
RottenTomatoes-IMDB (RI)	Movies	600	190	3
Books2 (B2)	Books	394	92	9
WDC-Computers (CO)	Product	1,100	300	2
WDC-Cameras (CA)	Product	1,100	300	2
WDC-Watches(WT)	Product	1,100	300	2
WDC-Shoes (SH)	Product	1,100	300	2

Evaluation metric. Following most ER works [21, 42, 49], we evaluate the above prediction results using F1 score, which is harmonic mean of precision and recall for the *matching* pairs. Specifically, let $\mathtt{TP}$ denote true positives, which are matching pairs correctly outputted by Matcher $\mathcal{M}$. Let $\mathtt{FP}$ denote false positives, which are non-matching pairs incorrectly outputted by $\mathcal{M}$. Let $\mathtt{FN}$ denote false negatives, which are matching pairs omitted by $\mathcal{M}$. Then, we can respectively compute precision and recall as $\mathtt{P} = \mathtt{TP}/(\mathtt{TP} + \mathtt{FP})$ and $\mathtt{R} = \mathtt{TP}/(\mathtt{TP} + \mathtt{FN})$. Based on this, we can compute F1 score as $\mathtt{F1} = 2 \cdot \mathtt{P} \cdot \mathtt{R}/(\mathtt{P} + \mathtt{R})$.

Comparison approaches. We present the implementation details of the approaches compared in our experiments as follows.

(1) DA: We implement the design choices in Table 1 for Feature Extractor $\mathcal{F}$, Feature Aligner $\mathcal{A}$ and Matcher $\mathcal{M}$. For implementing RNN for $\mathcal{F}$, we use the Hybrid model in DeepMatcher [49] implemented by a bidirectional RNN. For implementing LMs for $\mathcal{F}$, following Ditto [42], we use a Bert model [19] with 12 transformer layers and output a 768 dimensional feature vector. We set the max sequence length fed into Bert to be 256 (for WDC datasets) and 128 (for other datasets), as the text of WDC datasets is generally long. For implementing Matcher $\mathcal{M}$, we follow Ditto [42] to use one fully connected layer and a `Softmax` output layer. For implementing discrepancy-based Feature Aligner $\mathcal{A}$, we calculate MMD and K-order directly on the output layer of Feature Extractor $\mathcal{F}$ according to Equations (5) and (6). For adversarial-based Feature Aligner, we use one fully connected layer with `Sigmod` activation function in GRL, which is followed by three fully connected layers with `LeakyReLU` as activation function and a `Sigmod` layer for InvGAN and InvGAN+KD. Note that the hyper-parameters are chosen according to validation set, as mentioned above. For reconstruction-based Feature Aligner, we take a pre-trained model Bart [39] with its default settings to realize the reconstruction task in ED.

We use validation set $\mathcal{D}^{\mathsf{T}}_{\mathtt{val}}$ for choosing hyper-parameters. Specifically, for choosing β, we first give a candidate set of values

$\{0.001, 0.01, 0.1, 1, 5\}$, then run experiments for each value, and select the one with the highest F1 score on $\mathcal{D}^{\mathsf{T}}_{\mathsf{val}}$ for different datasets. Similarly, we set learning rate as 1e-5 or 1e-6, and batchsize as 32, on various datasets.

(2) NoDA: This is a baseline method without applying DA for ER. Specifically, it utilizes our **DADER** framework without Feature Aligner $\mathcal{A}$, where the $\mathcal{M}$ and $\mathcal{F}$ are totally the same with **DA**.

(3) Reweight: We consider an alternative solution of DA for ER [68], which is denoted as Reweight for ease of presentation. Different from **DADER** that focuses on feature learning, it aims at reweighting source entity pairs to emphasize the ones similar to the target. To this end, Reweight first uses word embedding techniques to convert an entity pair into a feature vector, and then measures similarity between source and target entity pairs in such feature space. Next, it computes weights of source entity pairs according to the obtained similarity scores, and trains Matcher $\mathcal{M}$ by considering the weights. We use the code provided from [68]. We use fastText to convert an entity pair into a 300 dimensional vector, which achieved the best performance. We run the default four machine learning classifiers for Matcher and report the best result.

(4) DeepMatcher: We use the best Hybrid method with the default settings in the original code [1]: the learning rate is 1e-3 by default, and the batch size is the same as DA.

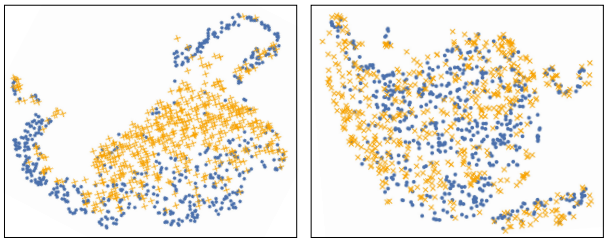
(5) Ditto: We run its original code from [2], which uses the pre-trained Roberta [44] model and three optimization operators by default, together with learning rate 3e-5.

All the methods are implemented using PyTorch [51] and the Transformers library [75]. All the experiments are evaluated on a server with 4 CPU cores (Intel Xeon Gold 6138 CPU @ 2.00GHz), 4 NVIDIA RTX 24GB GPUs and 1024GB memory, and the version of Python is 3.6.5. All the experiments are repeated in three times and the average results as well as standard deviations are reported.

6.2 Overall Results of DA for ER

6.2.1 Overall Performance. As the effect of domain adaptation may depend on the degree of domain shift between source and target datasets, we consider the following two settings to provide a more comprehensive evaluation. (1) *Similar Domains:* We select datasets from the same domain (such as product and citation as shown in Table 2), *e.g.,* Walmart-Amazon → Abt-Buy. (2) *Different Domains:* We select datasets across domains to prepare source and target datasets, *e.g.,* RottenTomatoes-IMDB (movie) → Abt-Buy (product). Note that we use the Pre-trained LMs as default Feature Extractor, and will discuss other design choices in Section 6.4.

Similar domains: Table 3 shows that, when the source and target datasets are from the similar domain, DA can get obvious improvement. In this setting, although $\mathcal{D}^{\mathsf{S}}$ and $\mathcal{D}^{\mathsf{T}}$ describe entities in the similar domain, these entities may have various attributes or different textual styles in the same attribute, which may cause the domain shift effect. Due to this effect, NoDA trained only on $\mathcal{D}^{\mathsf{S}}$ may focus on the information (*i.e.,* attributes, values, text, etc.) that is only useful to the source, leading to performance degradation on target dataset $\mathcal{D}^{\mathsf{T}}$. The experimental results show that DA can help to adjust the model to utilize the information shared by $\mathcal{D}^{\mathsf{S}}$ and $\mathcal{D}^{\mathsf{T}}$, so as to reduce the domain shift.



(a) NoDA (b) DA (InvGAN+KD)

Figure 5: Visualization of the effect of DA for Abt-Buy → Walmart-Amazon. Distributions of source (yellow) and target (blue) are much closer after DA (b) than without DA (a).

Take Abt-Buy and Walmart-Amazon as an example. The datasets share two attributes (`Title`, `Price`), while Abt-Buy has one textual attribute `Description` and Walmart-Amazon has three structural attributes (`Category`, `Brand`, `Modelno`). Table 3 shows that DA can improve performance by 6.8 and 14.2 on the basis of NoDA for Walmart-Amazon → Abt-Buy and Abt-Buy → Walmart-Amazon respectively. This is because DA guides $\mathcal{F}$ and $\mathcal{M}$ to make full use of the shared attributes (`Title`, `Price`), instead of paying much attention to the specific attributes in the source. Another interesting example is DBLP-Scholar and DBLP-ACM which have the same attributes (`Title`, `Authors`, `Venue`, `Year`) and different textual styles in attribute `Authors`, *i.e.,* abbreviation in DBLP-Scholar (e.g. *"m stonebraker"*) and full name in DBLP-ACM (e.g. *"michael stonebraker"*). The result shows that DA can also help $\mathcal{F}$ and $\mathcal{M}$ to gain useful information for $\mathcal{D}^{\mathsf{T}}$ from $\mathcal{D}^{\mathsf{S}}$ in this case.

Different domains: As shown in Table 4, when source and target datasets are from different domains, DA can obtain more significant improvements, *i.e.,* from 11.0 to 43.9 on the basis of the low performance of NoDA. The main reason is that, the source and target datasets in this setting are quite different, *i.e.,* their attributes may be completely different and describe totally different contents. Thus, the effect of domain shift would be more significant than the previous setting, which results in low performance of NoDA. The gains of DA in this set of experiments show that DA is helpful on datasets with significant differences on data distribution.

In order to help us understand how DA reduces domain shift, we use t-SNE [71], a well-adopted method for visualizing high-dimensional feature distributions [25, 45], to map the features of source and target datasets into two-dimensional space. Due to space limitation, we only show the case of Abt-Buy → Walmart-Amazon. We show their distributions in Figure 5, where yellow crosses and blue points respectively represent source and target entity pairs. The left corresponds to NoDA (*i.e.,* Feature Extractor $\mathcal{F}$ is only trained on source), while the right corresponds to DA (*i.e.,* Feature Extractor $\mathcal{F}$ has been adapted with the InvGAN+KD method in DA). We can see that source and target entity pairs are obviously more mixed when we apply DA. This results show that DA can guide $\mathcal{F}$ to make feature distributions of source and target much closer, which helps $\mathcal{M}$ make correct prediction on target $\mathcal{D}^{\mathsf{T}}$.

Finding 1: DA works well for the ER problem both on the datasets from similar domains (Table 3) and the datasets from different domains (Table 4).

Table 3: Similar domains: overall result of domain adaptation for entity resolution datasets, where Δ $F1$ denotes the $F1$ improvement achieved by the best DA method compared with the NoDA method.

Datasets		NoDA	Discrepancy-based		Adversarial-based			Reconstruction-based	Δ $F1$
Source	Target		MMD	K-order	GRL	InvGAN	InvGAN+KD	ED	
Walmart-Amazon	Abt-Buy	65.8 ± 4.6	**72.6 ± 3.0**	68.3 ± 0.3	68.4 ± 2.2	56.0 ± 31.7	69.6 ± 4.6	39.4 ± 8.4	6.8
Abt-Buy	Walmart-Amazon	56.9 ± 3.4	**71.1 ± 2.1**	62.0 ± 2.6	66.3 ± 0.8	47.5 ± 25.2	63.5 ± 3.1	45.7 ± 2.5	14.2
DBLP-Scholar	DBLP-ACM	**97.2 ± 0.1**	**97.2 ± 0.1**	96.2 ± 1.0	96.9 ± 0.1	97.1 ± 0.4	**97.2 ± 0.0**	96.8 ± 0.1	0.0
DBLP-ACM	DBLP-Scholar	77.8 ± 6.0	91.5 ± 0.2	88.9 ± 1.6	84.2 ± 3.9	92.1 ± 0.6	**92.3 ± 0.6**	90.5 ± 0.6	14.5
Zomato-Yelp	Fodors-Zagats	85.4 ± 16.2	92.2 ± 4.8	87.7 ± 4.3	89.1 ± 4.4	**94.5 ± 1.8**	93.5 ± 2.5	78.0 ± 5.1	9.1
Fodors-Zagats	Zomato-Yelp	47.6 ± 33.6	64.5 ± 6.3	72.6 ± 5.2	49.6 ± 28.2	29.7 ± 18.9	**75.0 ± 12.7**	0.0 ± 0.0	27.4

Table 4: Different domains: overall result of domain adaptation for entity resolution datasets, where Δ $F1$ denotes the $F1$ improvement achieved by the best DA method compared with the NoDA method.

Datasets		NoDA	Discrepancy-based		Adversarial-based			Reconstruction-based	Δ $F1$
Source	Target		MMD	K-order	GRL	InvGAN	InvGAN+KD	ED	
RottenTomatoes-IMDB	Abt-Buy	40.6 ± 12.0	43.6 ± 10.2	41.4 ± 11.8	42.7 ± 17.8	23.8 ± 6.1	**53.9 ± 5.9**	13.8 ± 7.7	13.3
RottenTomatoes-IMDB	Walmart-Amazon	38.4 ± 9.0	41.5 ± 8.4	41.9 ± 8.0	49.0 ± 8.8	35.1 ± 3.4	**49.4 ± 2.1**	30.7 ± 3.3	11.0
iTunes-Amazon	DBLP-ACM	80.3 ± 8.4	**94.5 ± 0.1**	86.9 ± 4.8	92.1 ± 2.1	57.7 ± 46.9	94.4 ± 0.6	77.5 ± 5.5	14.1
iTunes-Amazon	DBLP-Scholar	68.2 ± 9.8	86.9 ± 2.7	80.4 ± 6.2	85.4 ± 5.8	59.6 ± 47.7	**89.1 ± 0.7**	42.8 ± 11.7	20.9
Book2	Fodors-Zagats	49.6 ± 9.3	91.5 ± 0.8	78.2 ± 11.4	84.2 ± 7.1	**93.5 ± 2.2**	93.4 ± 1.7	78.1 ± 17.4	43.9
Book2	Zomato-Yelp	67.4 ± 3.3	73.0 ± 19.1	68.0 ± 21.0	54.0 ± 24.3	63.3 ± 10.7	**81.8 ± 10.0**	19.7 ± 14.0	14.4

Table 5: Results for WDC: Similar domains - different categories within the same website.

Datasets		NoDA	Discrepancy-based		Adversarial-based			Reconstruction-based	Δ $F1$
Source	Target		MMD	K-order	GRL	InvGAN	InvGAN+KD	ED	
computers	watches	**88.6 ± 1.4**	83.2 ± 2.3	87.1 ± 0.4	86.7 ± 1.6	86.2 ± 3.0	86.4 ± 1.1	76.5 ± 4.0	-1.5
watches	computers	82.1 ± 0.5	**85.6 ± 0.6**	82.9 ± 1.1	83.3 ± 0.8	80.6 ± 6.0	84.6 ± 0.6	64.9 ± 0.9	3.5
cameras	watches	87.1 ± 1.9	84.2 ± 1.7	86.0 ± 1.6	84.3 ± 2.1	85.9 ± 3.0	**88.3 ± 0.3**	68.5 ± 10.8	1.2
watches	cameras	86.1 ± 0.9	86.0 ± 0.2	85.4 ± 1.1	**86.7 ± 0.7**	85.2 ± 2.9	83.9 ± 3.1	71.3 ± 4.9	0.6
shoes	watches	83.6 ± 1.6	83.2 ± 1.2	82.6 ± 2.4	**84.2 ± 1.8**	83.3 ± 0.3	83.5 ± 3.2	69.7 ± 9.3	0.6
watches	shoes	76.3 ± 1.5	74.7 ± 2.3	76.9 ± 0.3	76.5 ± 2.6	74.0 ± 1.7	**77.0 ± 4.4**	65.7 ± 4.2	0.7
computers	shoes	71.6 ± 3.6	75.2 ± 2.3	74.5 ± 2.3	76.3 ± 2.2	72.9 ± 3.8	**76.5 ± 3.2**	62.1 ± 2.7	4.9
shoes	computers	83.3 ± 2.1	**85.8 ± 0.9**	83.7 ± 2.6	83.8 ± 2.1	85.0 ± 1.9	82.3 ± 2.6	58.7 ± 2.4	2.5
cameras	shoes	74.0 ± 2.6	65.5 ± 2.9	**77.6 ± 0.1**	76.9 ± 1.0	74.7 ± 3.7	76.5 ± 0.3	58.6 ± 6.9	3.6
shoes	cameras	79.4 ± 0.8	81.9 ± 6.3	82.0 ± 1.8	83.2 ± 4.1	85.0 ± 1.4	**87.6 ± 1.0**	69.5 ± 4.4	8.3
computers	cameras	83.9 ± 3.5	84.0 ± 1.2	85.7 ± 0.8	84.3 ± 0.9	85.6 ± 2.4	**86.7 ± 1.7**	75.5 ± 2.9	2.8
cameras	computers	87.0 ± 1.7	**88.0 ± 0.2**	87.1 ± 1.7	87.2 ± 1.3	86.4 ± 0.2	87.8 ± 1.3	71.9 ± 3.0	1.1

We notice that the improvement of DA is not always significant. For example, as shown in Table 3, for DBLP-Scholar → DBLP-ACM, DA achieves the same results as NoDA. The main reason is that the models originally trained on source $\mathcal{D}^S$ already predict target $\mathcal{D}^T$ well. To further investigate this, we evaluate the WDC datasets in Table 5, and the results show that the gain of DA is not obvious (from −1.5 to 8.3). This is because the four WDC datasets, although correspond to different categories, have a same textual attribute (*i.e.,* `Title`) that follows the same word vocabulary. Thus, the data distribution of different WDC datasets would be very similar. This can be validated by some results of NoDA on target $\mathcal{D}^T$ (*e.g.,* computers → watches: 88.6, watches → cameras: 86.1, watches → shoes: 76.3 and cameras → computers: 87.0), which are better than that of the state-of-the-art ER model [42] trained on their own training sets (computers: 80.8, cameras: 80.9, watches: 85.1, and shoes: 75.9). The results indicate that the domain shift may not be significant in this case, and thus there is little space for DA to improve.

6.2.2 In-Depth Analysis. The above finding inspires us to conduct an in-depth analysis to examine whether the "distance" between source and target affects the DA performance. We calculate the distance between source and target datasets by using maximum mean discrepancy (MMD) [30], as discussed in Section 5. Specifically, for both source and target, we use a pre-trained Bert [19] as Feature Extractor to obtain a 768-dimensional feature space. The smaller the MMD is, the closer distance the datasets have. The result is shown in Figure 6. Given a same target dataset $\mathcal{D}^T$, we can clearly observe that DA achieves higher F1 scores when the source and target are closer to each other. This indicates a possibility of using such distance to select better source datasets for a given target, which will be studied in the future work.

Finding 2: DA achieves higher F1 scores when the source dataset is closer to the target dataset, which identifies a possible research direction on source data selection, *e.g.,* choosing a "close" domain for DA to improve the performance.

6.3 Evaluation on Feature Aligner $\mathcal{A}$

This section explores the benefits and limitations of the design choices for Feature Aligner $\mathcal{A}$ shown in Table 1. Specifically, we

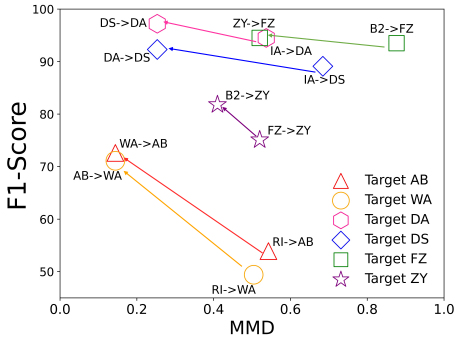


Figure 6: Evaluation on distance (MMD) between source and target and the DA performance. The result shows that, given a same target dataset, DA achieves higher F1 scores when the source and target have a smaller MMD distance.

analyze the *successful cases* where $\mathcal{A}$ improves the ER performance, and the *failure cases* where $\mathcal{A}$ does not.

6.3.1 Successful Cases Analysis. As observed from Tables 3 and 4, MMD and InvGAN+KD outperform NoDA in nearly all the cases, while these two methods seem to be much of a muchness. To further compare them, we investigate their training epochs with different learning rates by using Books2 as source and Fodors-Zagats as target. Figure 7 shows that MMD can always be convergent with enough epochs, while InvGAN+KD is oscillate. When the learning rate is 1e-5, NoDA and MMD converge to F1 scores 25 and 78 respectively, while InvGAN+KD first reaches a high F1 score at epoch 6 and then drops sharply. With a smaller learning rate, *e.g.,* 1e-7 in Figure 7(c), InvGAN+KD does not vibrate violently and has a convergence trend. However, the epoch corresponding to the best F1 score changes from 6 to 12. It is reasonable, smaller learning rate, longer training time.

Finding 3: Discrepancy-based DA performs well to be convergent with enough training epochs and achieves obvious improvements, while adversarial-based DA may be oscillate. The oscillation may be reduced by reducing learning rate, which may lead to more training epochs.

The above finding implies that InvGAN+KD is sensitive to the hyper-parameters (*e.g.,* learning rate), and thus a small validation set is necessary to InvGAN+KD. If the researcher has small labeled target validation set or enough experience and time to adjust the hyper-parameters, we suggest a more promising adversarial-based method (InvGAN+KD). On the contrary, it is suggested to use the discrepancy-based method with stable performance.

6.3.2 Failure Cases Analysis. We observe from Tables 3 and 4 that some methods are worse than NoDA. Thus, we provide an in-depth analysis to discuss why the methods fail in some cases.

InvGAN. We can see that InvGAN is worse than NoDA in many cases. For example, for Fodors-Zagats → Zomato-Yelp, the result of InvGAN is 29.7 while NoDA achieves 47.6. To investigate the reason, we examine the training epochs of InvGAN for Fodors-Zagats → Zomato-Yelp. As shown in Figure 8, with the increase of training epochs, the F1 scores of both source and target drop sharply and become very oscillate. The main reason is that, as discussed in Section 5.2, InvGAN only adjusts Feature Extractor to make target features $\mathbf{x}^T$ as similar to source features $\mathbf{x}^S$ as possible, while ignoring whether the adjusted features are discriminative or not. Thus, the obtained features may damage the performance of $\mathcal{M}$. This can be reflected by the result that, after the adaptation, $\mathcal{M}$ becomes dramatically worse even on the source dataset. On the other hand, we also observe from Figure 8 that InvGAN+KD can effectively address this problem to achieve much higher and more stable performance. This is attributed to the knowledge distillation used in InvGAN+KD for retaining the classification ability of $\mathcal{M}$.

ED. The reconstruction-based method ED also achieves inferior performance in almost all the cases, which is quite different from the case of DA for CV [27, 73]. This is attributed to the encoder-decoder approach that is hard to capture and reconstruct the textual information of original entity pairs. This result shows that the reconstruction-based method may not be readily applicable to ER.

GRL. GRL is generally good, while failing to outperform NoDA in a few cases, *e.g.,* Book2 → Zomato-Yelp. This is because the GRL training is usually not stable, as pointed out by many existing studies [9, 34]. During training, the confrontation between $\mathcal{A}$ and $\mathcal{F}$ may lead to the failure of convergence of $\mathcal{F}$ and $\mathcal{M}$. Either the selection of hyper-parameters or the initialization of model parameters may have a great impact on the final results.

Finding 4: Not all successful DA methods in CV and NLP are applicable to ER. It is necessary to ensure that the learned features satisfy both domain-invariant and discriminative characteristics, while the balance between these two factors is the key for DA for ER.

6.4 Evaluation on Feature Extractor $\mathcal{F}$

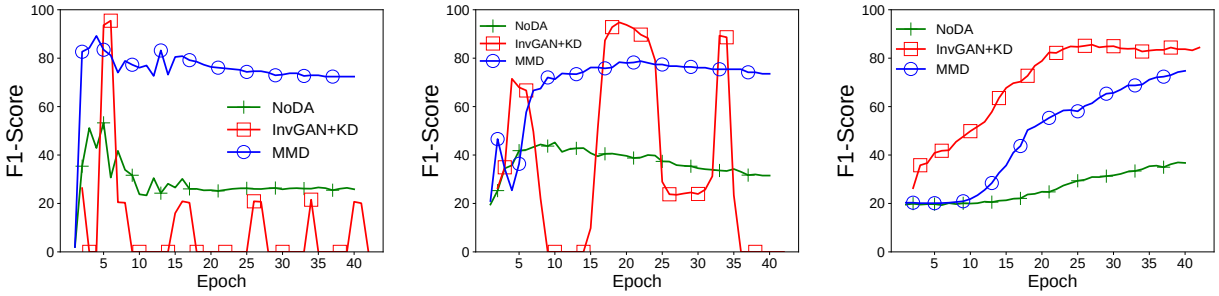
After using Pre-trained LMs (*i.e.,* Bert) as default $\mathcal{F}$ in the previous sections, we next explore the performance of using other choices. We examine a bidirectional RNN, because, as reported in DeepMatcher [49], a hybrid method using bidirectional RNN with decomposable attention achieves the best ER performance. Thus, in this paper, we use the same hybrid architecture as the Feature Extractor based on its code [1]. For dealing with the situation that $\mathcal{D}^S$ and $\mathcal{D}^T$ have different attributes, we learn a universal RNN, instead of learning a separate RNN for each attribute, following the existing work DTAL [35]. Figure 9 shows that Bert is better than RNN in all of the three dataset groups. The F1 scores are lower and the improvements of DA are not obvious with RNN as Feature Extractor. This suggests that the transfer ability of RNN is weak, so the model trained on source $\mathcal{D}^S$ relies heavily on itself, and can not work well on target $\mathcal{D}^T$. As a result, it is hard for DA to adjust RNN to extract domain-invariant features.

Finding 5: When Feature Extractor is a bidirectional RNN, the gains of DA are not obvious. The improvement of DA for ER depends on the transferability of Pre-trained LMs.

6.5 Comparison with Existing Approaches

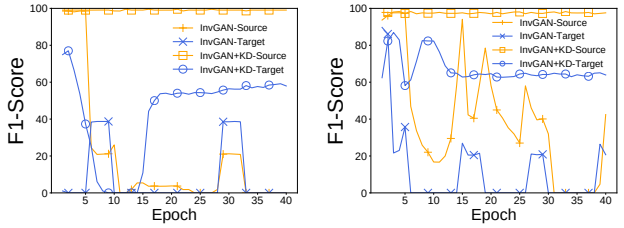
This section compares our **DADER** approach with existing approaches. Note that we use the result of InvGAN+KD, which achieves the best performance under the **DADER** framework.

6.5.1 Comparison among Alternative DA Solutions. We first compare **DADER** with the alternative DA for ER approach Reweight,



(a) Learning Rate = 1e-5 (b) Learning Rate = 1e-6 (c) Learning Rate = 1e-7

Figure 7: Convergence comparison of MMD and InvGAN+KD. Each figure shows the changes of F1 score with epochs. The red line corresponds to InvGAN+KD, the blue line corresponds to MMD, and the green line corresponds to NoDA.



(a) Fodors-Zagats → Zomato-Yelp (b) Zomato-Yelp → Fodors-Zagats

Figure 8: The F1 score curves of InvGAN and InvGAN+KD on source and target datasets.

which is described in Section 6.1. Figure 10 shows that **DADER** is much better than Reweight. This is attributed to the different domain adaptation strategies used in these two approaches. Reweight focuses on an *instance-level* approach that reweights source entity pairs and adapts them to be alike the target. This approach may not be effective when the similarity between source and target is hard to measure or there is a small amount of source data. In contrast, **DADER** adopts a *feature-level* approach that learns domain-invariant and discriminative features, which has been shown to be more promising in many applications, such as CV and NLP.

Finding 6: Feature-level DA approaches that learn domain-invariant and discriminative features are better than instance-based approaches that reweight source data.

6.5.2 Comparison with Using Some Target Labels. We explore the performance of different methods with some labeled target data. For fair comparison, the target datasets are the original experimental datasets [49] provided by DeepMatcher, which are divided into train, valid and test by 3:1:1. We select the labeled data from the train set and record the F1 score on the test set. Due to the limitation of space, we report four datasets with large amount of data in Figure 11. The green and red lines correspond to NoDA and InvGAN+KD respectively. In addition, we compare our method with Ditto [42] (*i.e.,* blue line) and DeepMatcher [49] (*i.e.,* brown line). We select the labeled data according to the maximum entropy principle, which is the basic principle in active learning. We label 200 of the training set per round, with a total of 4 rounds.

When the number of labeled data is small, InvGAN+KD is obviously better than other methods. With the increase of the label number, Ditto and DeepMacher get improved, but the overall performance of DeepMacher is lower than other methods. The results show that, to achieve good performance, DeepMacher needs more labeled data, and Ditto can perform well with a few labeled data. However, **DADER** gets the highest F1 scores in various cases.

Finding 7: The performance of the model after DA can always be maintained at a high level with some labeled data, while outperforming state-of-the-art DL-based ER methods, DeepMatcher [49] and Ditto [42].

7 RELATED WORK

Entity resolution. There have been rule-based methods [23, 58], matching functions [4, 5, 20, 60], crowd-based methods [11, 12, 17, 22, 40, 41, 76] and traditional ML-based models [3, 15, 47, 66] for ER (see the book [50] for more details). Recently, DL-based methods have been widely used in ER, and achieved the state-of-the-art results. DeepER [21] designs two deep neural networks to extract features of entity pairs, and models ER as a binary classification task. DeepMatcher [49] systematically defines the architecture and design space of DL solutions for ER. Ditto [42] first applies pre-trained language models to ER, which can reduce the number of training data needed. Even though, a lot of training entity pairs are still needed for DL-based methods.

Domain adaptation. Domain adaptation (DA) [25, 43, 45, 64, 69], which is a case of transfer learning [36, 80], is an effective way to reuse labeled source data to (possibly) different target data. Existing DA solutions can be broadly categorized into *instance-level*, *feature-level* and others. (1) Instance-level: these methods aim at reusing source data instances by adapting them to target distribution. To this end, traditional studies reweight source data instances to emphasize the ones similar to the target [8, 16]. Recent DL methods are proposed to learn a mapping function to adapt source data instances to be alike the target [13, 33], or generate pseudo labels for target data [26]. (2) Feature-level: these methods focus on learning *domain-invariant* and *discriminative* features. Existing feature-level methods can be divided into three categories. Discrepancy-based methods utilize various metrics, *e.g.,* MMD [45, 46, 70], second-order statistics [61, 62] and higher-order moment [78], to compute and reduce distribution discrepancy between source and target. Adversarial-based methods leverage the adversarial learning framework, *e.g.,* gradient reverse [25] and GAN-based minimax training [69] to learn domain-invariant features, which is followed by many studies [6, 10, 24, 37, 48, 53]. Reconstruction-based methods introduce data reconstruction as auxiliary task to boost the feature learning process [7, 27, 79]. (3) Others: there are many other DA studies, such as meta-learning with DA dealing with multiple

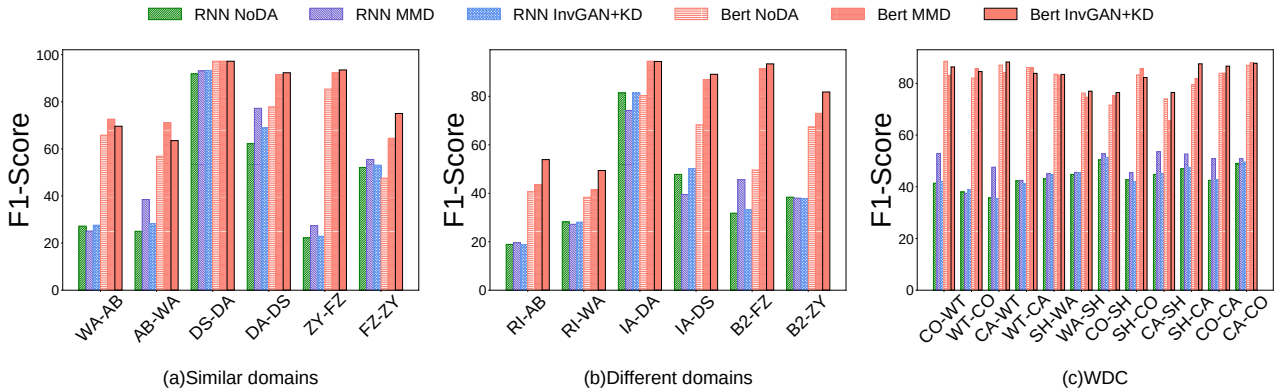


Figure 9: The performance comparison of Pre-trained LMs and RNN. Each dataset has six bars, where the left three take RNN as $\mathcal{F}$ and the right three take Bert as $\mathcal{F}$.

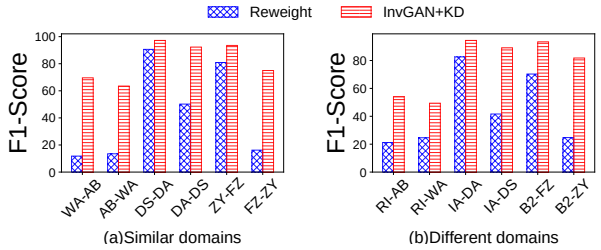


Figure 10: Comparison between DADER and Reweight.

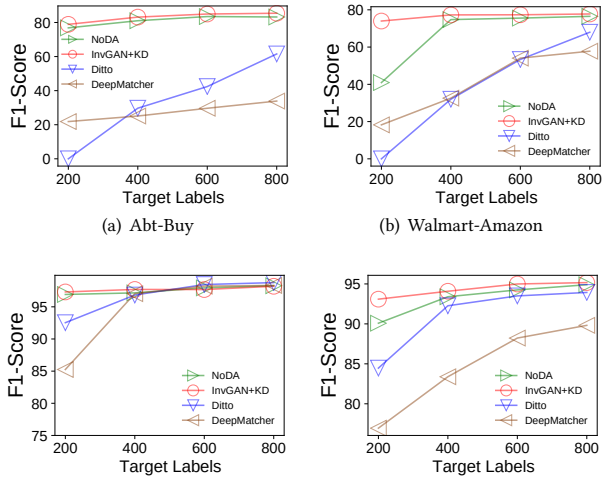


Figure 11: Comparison with using some target labels.

tasks [54] and multi-step DA [14, 63]. Moreover, many pretraining-based solutions are proposed, such as selecting source pretraining data to benefit the target [18] and performing pretraining task on the target [31]. Among the above DA approaches, we select the feature-level methods, which are the most fruitful and successful ones (see recent surveys [73, 74]), and examine which DA design choices are helpful for ER, the core data integration task.

Domain adaptation for entity resolution. Due to the good performance of DA in CV and NLP, there are some attempts to apply DA to ER. Thirumuruganathan et al. introduce an instance-level method to reweight source data instances and make them adaptable for the target [68]. Kasai et al. apply DA to ER with gradient reverse [35], which is one special case in our proposed design space. However, there is no research to systematically study DA for ER, especially comparing various DA solutions under the same framework. Thus, it is hard for practitioners to understand benefits and limitations of applying DA to ER.

8 CONCLUSIONS

In this paper we have advanced the SOTA in applying DA to ER, by defining a large space of DA solutions, and developing six representative methods for Feature Aligner. We have conducted extensive experiments to evaluate different combinations of the methods in the design space (Section 6) with insightful empirical findings. DA can significantly improve ER when the model trained on the source does not perform well on the target (**Finding 1**), and may be more beneficial when choosing a "close" source domain for DA (**Finding 2**). For the Feature Aligner methods, discrepancy-based methods (*e.g.,* MMD) are more stable but adversarial-based method InvGAN+KD works the best when carefully tuning hyper-parameters (**Finding 3**). Moreover, not all successful DA methods in CV and NLP are suitable to ER (**Finding 4**) and pre-trained LM Feature Extractor works better than RNN based methods (**Finding 5**). Although there may be different methods for DA for ER, feature-level DA approaches that learn domain-invariant and discriminative features are better than instance-based approaches (**Finding 6**). Last, when the labels for target datasets are available, using DA requires much less training data from the target to achieve a high accuracy, comparing with the SOTA DL based methods with/without pre-trained languages models (**Finding 7**).

This research has thrown up several questions in need of further investigation. For example, whether DA using multiple labeled source data can further help ER? If so, shall we use them all or a subset of source datasets? How to combine the **DADER** framework with the ER blocking to build a scalable ER framework?

Acknowledgement. This work was partly supported by NSF of China (62122090, 62072461, 61925205, 62102215, U1911203), Huawei, TAL education, Beijing National Research Center for Information Science and Technology (BNRist), China National Postdoctoral Program for Innovative Talents (BX2021155), China Postdoctoral Science Foundation(2021M691784), Shuimu Tsinghua Scholar and Zhejiang Lab's International Talent Fund for Young Professionals.

REFERENCES

[1] 2018. *Code of DeepMatcher.* https://github.com/anhaidgroup/deepmatcher

[2] 2020. *Code of Ditto.* https://github.com/megagonlabs/ditto

[3] Fabio Azzalini, Songle Jin, Marco Renzi, and Letizia Tanca. 2021. Blocking techniques for entity linkage: A semantics-based approach. *Data Science and Engineering* 6, 1 (2021), 20–38.

[4] Omar Benjelloun, Hector Garcia-Molina, David Menestrina, Qi Su, Steven Euijong Whang, and Jennifer Widom. 2009. Swoosh: a generic approach to entity resolution. *The VLDB Journal* 18, 1 (2009), 255–276.

[5] Mikhail Bilenko and Raymond J Mooney. 2003. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. 39–48.

[6] Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, Yunfei Bai, Matthew Kelcey, Mrinal Kalakrishnan, Laura Downs, Julian Ibarz, Peter Pastor, Kurt Konolige, et al. 2018. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 4243–4250.

[7] Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. 2016. Domain separation networks. *arXiv preprint arXiv:1608.06019* (2016).

[8] Lorenzo Bruzzone and Mattia Marconcini. 2009. Domain adaptation problems: A DASVM classification technique and a circular validation strategy. *IEEE transactions on pattern analysis and machine intelligence* 32, 5 (2009), 770–787.

[9] Guanyu Cai, Yuqin Wang, Lianghua He, and MengChu Zhou. 2019. Unsupervised domain adaptation with adversarial residual transform networks. *IEEE transactions on neural networks and learning systems* 31, 8 (2019), 3073–3086.

[10] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Michael I Jordan. 2018. Partial transfer learning with selective adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2724–2732.

[11] Chengliang Chai, Guoliang Li, Jian Li, Dong Deng, and Jianhua Feng. 2016. Cost-effective crowdsourced entity resolution: A partial-order approach. In *Proceedings of the 2016 International Conference on Management of Data*. 969–984.

[12] Chengliang Chai, Guoliang Li, Jian Li, Dong Deng, and Jianhua Feng. 2018. A partial-order-based framework for cost-effective crowdsourced entity resolution. *The VLDB Journal* 27, 6 (2018), 745–770.

[13] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. 2018. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 8789–8797.

[14] Sumit Chopra, Suhrid Balakrishnan, and Raghuraman Gopalan. 2013. Dlid: Deep learning for domain adaptation by interpolating between domains. In *ICML workshop on challenges in representation learning*, Vol. 2. Citeseer.

[15] Peter Christen. 2008. Automatic record linkage using seeded nearest neighbour and support vector machine classification. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 151–159.

[16] Wen-Sheng Chu, Fernando De la Torre, and Jeffery F Cohn. 2013. Selective transfer machine for personalized facial action unit detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3515–3522.

[17] Lizhen Cui, Jing Chen, Wei He, Hui Li, Wei Guo, and Zhiyuan Su. 2021. Achieving Approximate Global Optimization of Truth Inference for Crowdsourcing Microtasks. *Data Science and Engineering* 6, 3 (2021), 294–309.

[18] Xiang Dai, Sarvnaz Karimi, Ben Hachey, and Cecile Paris. 2019. Using similarity measures to select pretraining data for NER. *arXiv preprint arXiv:1904.00585* (2019).

[19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[20] AnHai Doan, Pradap Konda, Paul Suganthan GC, Yash Govind, Derek Paulsen, Kaushik Chandrasekhar, Philip Martinkus, and Matthew Christie. 2020. Magellan: toward building ecosystems of entity matching solutions. *Commun. ACM* 63, 8 (2020), 83–91.

[21] Muhammad Ebraheem, Saravanan Thirumuruganathan, Shafiq Joty, Mourad Ouzzani, and Nan Tang. 2018. Distributed representations of tuples for entity resolution. *Proceedings of the VLDB Endowment* 11, 11 (2018), 1454–1467.

[22] Ju Fan, Guoliang Li, Beng Chin Ooi, Kian-lee Tan, and Jianhua Feng. 2015. icrowd: An adaptive crowdsourcing framework. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*. 1015–1030.

[23] Wenfei Fan, Xibei Jia, Jianzhong Li, and Shuai Ma. 2009. Reasoning about record matching rules. *Proceedings of the VLDB Endowment* 2, 1 (2009), 407–418.

[24] Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*. PMLR, 1180–1189.

[25] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The journal of machine learning research* 17, 1 (2016), 2096–2030.

[26] Yixiao Ge, Dapeng Chen, and Hongsheng Li. 2020. Mutual mean-teaching: Pseudo label refinery for unsupervised domain adaptation on person re-identification. *arXiv preprint arXiv:2001.01526* (2020).

[27] Muhammad Ghifary, W Bastiaan Kleijn, Mengjie Zhang, David Balduzzi, and Wen Li. 2016. Deep reconstruction-classification networks for unsupervised domain adaptation. In *European Conference on Computer Vision*. Springer, 597–613.

[28] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial networks. *arXiv preprint arXiv:1406.2661* (2014).

[29] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. 2012. A kernel two-sample test. *The Journal of Machine Learning Research* 13, 1 (2012), 723–773.

[30] Han Guo, Ramakanth Pasunuru, and Mohit Bansal. 2020. Multi-source domain adaptation for text classification via distancenet-bandits. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 7830–7838.

[31] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. 2020. Don't stop pretraining: adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964* (2020).

[32] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[33] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. 2018. Cycada: Cycle-consistent adversarial domain adaptation. In *International conference on machine learning*. PMLR, 1989–1998.

[34] Constantinos Karouzos, Georgios Paraskevopoulos, and Alexandros Potamianos. 2021. UDALM: Unsupervised Domain Adaptation through Language Modeling. *arXiv preprint arXiv:2104.07078* (2021).

[35] Jungo Kasai, Kun Qian, Sairam Gurajada, Yunyao Li, and Lucian Popa. 2019. Low-resource deep entity resolution with transfer and active learning. *arXiv preprint arXiv:1906.08042* (2019).

[36] Atsutoshi Kumagai, Tomoharu Iwata, and Yasuhiro Fujiwara. 2020. Transfer metric learning for unseen domains. *Data Science and Engineering* 5, 2 (2020), 140–151.

[37] Abhishek Kumar, Prasanna Sattigeri, Kahini Wadhawan, Leonid Karlinsky, Rogerio Feris, William T Freeman, and Gregory Wornell. 2018. Co-regularized alignment for unsupervised domain adaptation. *arXiv preprint arXiv:1811.05443* (2018).

[38] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436–444.

[39] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461* (2019).

[40] Guoliang Li, Chengliang Chai, Ju Fan, Xueping Weng, Jian Li, Yudian Zheng, Yuanbing Li, Xiang Yu, Xiaohang Zhang, and Haitao Yuan. 2017. CDB: optimizing queries with crowd-based selections and joins. In *Proceedings of the 2017 ACM International Conference on Management of Data*. 1463–1478.

[41] Guoliang Li, Chengliang Chai, Ju Fan, Xueping Weng, Jian Li, Yudian Zheng, Yuanbing Li, Xiang Yu, Xiaohang Zhang, and Haitao Yuan. 2018. CDB: A crowd-powered database system. *Proceedings of the VLDB Endowment* 11, 12 (2018), 1926–1929.

[42] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2020. Deep entity matching with pre-trained language models. *arXiv preprint arXiv:2004.00584* (2020).

[43] Tongyu Liu, Ju Fan, Yinqing Luo, Nan Tang, Guoliang Li, and Xiaoyong Du. 2021. Adaptive data augmentation for supervised learning over missing data. *Proceedings of the VLDB Endowment* 14, 7 (2021), 1202–1214.

[44] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[45] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. 2015. Learning transferable features with deep adaptation networks. In *International conference on machine learning*. PMLR, 97–105.

[46] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. 2017. Deep transfer learning with joint adaptation networks. In *International conference on machine learning*. PMLR, 2208–2217.

[47] Andrew McCallum and Ben Wellner. 2004. Conditional models of identity uncertainty with application to noun coreference. *Advances in neural information processing systems* 17 (2004), 905–912.

[48] Saeid Motiian, Quinn Jones, Seyed Mehdi Iranmanesh, and Gianfranco Doretto. 2017. Few-shot adversarial domain adaptation. *arXiv preprint arXiv:1711.02536* (2017).

[49] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. 2018. Deep learning for entity matching: A design space exploration. In *Proceedings of the 2018 International Conference on Management of Data*. 19–34.

[50] George Papadakis, Ekaterini Ioannou, Emanouil Thanos, and Themis Palpanas. 2021. The Four Generations of Entity Resolution. *Synthesis Lectures on Data Management* 16, 2 (2021), 1–170.

[51] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019), 8026–8037.

[52] Anna Primpeli, Ralph Peeters, and Christian Bizer. 2019. The WDC training dataset and gold standard for large-scale product matching. In *Companion Proceedings of The 2019 World Wide Web Conference*. 381–386.

[53] Minho Ryu and Kichun Lee. 2020. Knowledge Distillation for BERT Unsupervised Domain Adaptation. *arXiv preprint arXiv:2010.11478* (2020).

[54] Doyen Sahoo, Hung Le, Chenghao Liu, and Steven CH Hoi. 2018. Meta-learning with domain adaptation for few-shot learning under domain shift. (2018). https://openreview.net/pdf?id=ByGOuo0cYm

[55] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* (2019).

[56] Iulian Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 31.

[57] Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018. Disan: Directional self-attention network for rnn/cnn-free language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[58] Rohit Singh, Vamsi Meduri, Ahmed Elmagarmid, Samuel Madden, Paolo Papotti, Jorge-Arnulfo Quiané-Ruiz, Armando Solar-Lezama, and Nan Tang. 2017. Generating concise entity matching rules. In *Proceedings of the 2017 ACM International Conference on Management of Data*. 1635–1638.

[59] Rohit Singh, Venkata Vamsikrishna Meduri, Ahmed Elmagarmid, Samuel Madden, Paolo Papotti, Jorge-Arnulfo Quiané-Ruiz, Armando Solar-Lezama, and Nan Tang. 2017. Synthesizing entity matching rules by examples. *Proceedings of the VLDB Endowment* 11, 2 (2017), 189–202.

[60] Parag Singla and Pedro Domingos. 2006. Entity resolution with markov logic. In *Sixth International Conference on Data Mining (ICDM'06)*. IEEE, 572–582.

[61] Baochen Sun, Jiashi Feng, and Kate Saenko. 2016. Return of frustratingly easy domain adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 30.

[62] Baochen Sun and Kate Saenko. 2016. Deep coral: Correlation alignment for deep domain adaptation. In *European conference on computer vision*. Springer, 443–450.

[63] Ben Tan, Yu Zhang, Sinno Pan, and Qiang Yang. 2017. Distant domain transfer learning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 31.

[64] Hui Tang and Kui Jia. 2020. Discriminative adversarial domain adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 5940–5947.

[65] Nan Tang, Ju Fan, Fangyi Li, Jianhong Tu, Xiaoyong Du, Guoliang Li, Sam Madden, and Mourad Ouzzani. 2020. RPT: relational pre-trained transformer is almost all you need towards democratizing data preparation. *arXiv preprint arXiv:2012.02469* (2020).

[66] Sheila Tejada, Craig A Knoblock, and Steven Minton. 2002. Learning domain-independent string transformation weights for high accuracy object identification. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. 350–359.

[67] Saravanan Thirumuruganathan, Han Li, Nan Tang, Mourad Ouzzani, Yash Govind, Derek Paulsen, Glenn Fung, and AnHai Doan. 2021. Deep learning for blocking in entity matching: a design space exploration. *Proceedings of the VLDB Endowment* 14, 11 (2021), 2459–2472.

[68] Saravanan Thirumuruganathan, Shameem A Puthiya Parambath, Mourad Ouzzani, Nan Tang, and Shafiq Joty. 2018. Reuse and adaptation for entity resolution through transfer learning. *arXiv preprint arXiv:1809.11084* (2018).

[69] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. 2017. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7167–7176.

[70] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. 2014. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474* (2014).

[71] Laurens Van Der Maaten. 2013. Barnes-hut-sne. *arXiv preprint arXiv:1301.3342* (2013).

[72] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research* 11, 12 (2010).

[73] Mei Wang and Weihong Deng. 2018. Deep visual domain adaptation: A survey. *Neurocomputing* 312 (2018), 135–153.

[74] Garrett Wilson and Diane J Cook. 2020. A survey of unsupervised deep domain adaptation. *ACM Transactions on Intelligent Systems and Technology (TIST)* 11, 5 (2020), 1–46.

[75] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771* (2019).

[76] Jingru Yang, Ju Fan, Zhewei Wei, Guoliang Li, Tongyu Liu, and Xiaoyong Du. 2018. Cost-effective data annotation using game-based crowdsourcing. *Proceedings of the VLDB Endowment* 12, 1 (2018), 57–70.

[77] Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. 2017. Comparative study of CNN and RNN for natural language processing. *arXiv preprint arXiv:1702.01923* (2017).

[78] Werner Zellinger, Thomas Grubinger, Edwin Lughofer, Thomas Natschläger, and Susanne Saminger-Platz. 2017. Central moment discrepancy (cmd) for domain-invariant representation learning. *arXiv preprint arXiv:1702.08811* (2017).

[79] Fuzhen Zhuang, Xiaohu Cheng, Ping Luo, Sinno Jialin Pan, and Qing He. 2015. Supervised representation learning: Transfer learning with deep autoencoders. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

[80] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. 2020. A comprehensive survey on transfer learning. *Proc. IEEE* 109, 1 (2020), 43–76.