TransER: Homogeneous Transfer Learning for Entity Resolution

Nishadi Kirielle
nishadi.kirielle@anu.edu.au
The Australian National University
Canberra, ACT 2600, Australia

Peter Christen
peter.christen@anu.edu.au
The Australian National University
Canberra, ACT 2600, Australia

Thilina Ranbaduge
thilina.ranbaduge@anu.edu.au
The Australian National University
Canberra, ACT 2600, Australia

ABSTRACT

Entity resolution (ER) is the process of linking records that refer to the same entity across one or more databases. While recent advances in supervised learning can provide high quality results for ER, these often come with large efforts to obtain labelled training data. Transfer learning (TL) can overcome this expensive labelling task by utilising existing training data from a semantically related *source* domain to classify instances in a *target* domain where no training data are available. However, most existing TL solutions for ER involve deep learning models that have shown to be mostly useful for long textual and unstructured attributes. These models are less successful for short structured attributes such as personal data that are known to contain variations and typographical errors. In this paper, we propose a novel TL framework for resolving entities in structured data. We assume homogeneous domains that have the same feature space (same attribute types and similarity functions) for transferring. As records are sourced from different domains, there however can be three key challenges. The marginal probability distributions of the data in the two domains can be different, there can be feature vectors that have contradicting labels in the two domains resulting in different class conditional probability distributions, and the class imbalance and bi-modal data distributions common in ER make it challenging to apply existing TL methods. We address these challenges with three contributions: an instance selector to choose source instances with a high confidence and a similar local structure to the target domain, a label generator that creates pseudo labels for target instances, and a final classifier that labels target instances using only high confidence pseudo labels. On seven data sets we show that our framework outperforms the best of several state-of-the-art methods by up to 13% in precision and 50% in recall, while also being substantially faster.

1 INTRODUCTION

Entity Resolution (ER) is the process of linking records of the same entity across one or more databases in the absence of unique entity identifiers. Domains including health analytics, national censuses, crime and fraud detection, e-commerce, national security, and the social sciences have an increasingly high demand for ER to facilitate advanced analytics of integrated data [9, 17, 30, 33, 40].

The process of linking two databases is illustrated in Figure 1 [11, 46], where first the pre-processed databases are blocked to reduce the quadratic record pair comparison space, and then the attribute values of record pairs in each block are compared using similarity functions. This step generates a *feature vector* for each compared record pair that holds the similarity score of two attribute values as a feature value. For a record pair of two publications, its feature vector would, for example, include the textual similarities of publication titles, venues, and author names. In this context, we refer to such a feature vector as an *instance*. The feature vectors for all compared record pairs then generate a *feature matrix*. In the classification step, either supervised or unsupervised methods are employed to classify each record pair and assign it a *match* (referring to the same entity) or *non-match* (referring to different entities) class label [9]. Finally, the obtained results are evaluated with regard to quality and efficiency [22].

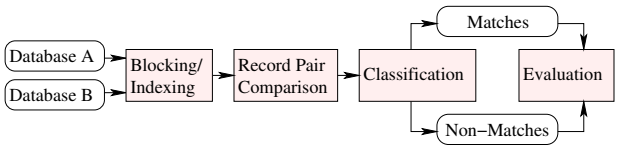

Figure 1: The general process of linking two databases.

Despite the existence of unsupervised ER approaches [9, 46], supervised ER methods have recently gained much attention due to the high quality results that can be obtained when sufficient training data are available. Supervised methods, however, often require extensive human efforts to label record pairs as matches and non-matches. Such manual labelling is costly and time consuming, making it infeasible in many real-world applications. A recent study has, for example, reported that a random forest based classifier required 1.5 million training labels to obtain a precision and recall of 99% each even for linking two fairly clean databases of records describing consumer products [16].

An intuitive approach to avoid this tedious labelling task is to utilise existing labelled training data from a related domain. A *domain* in ER, as we formally define in Section 3, consists of record pairs from two databases, and the similarities (features) calculated between them. In such an approach, we can use previously labelled and semantically related training data, which we refer to as the *source domain*, to classify record pairs from unlabelled databases in the *target domain*. For example, as shown in Table 1, assume we already have labels from the linked data sets DBLP and ACM and we aim to link DBLP with Scholar [31]. In this scenario the source domain consists of record pairs from DBLP and ACM while the target domain consists of record pairs from DBLP and Scholar.

However, supervised learning tasks generally perform well only if the training and testing data are drawn from the same feature space and furthermore have the same distribution [45]. The idea of utilising a source domain with a different distribution to label a target domain has attracted the attention of researchers and given rise to the research area of transfer learning (TL), which has provided promising results in a range of fields including computer vision [8, 20, 36, 52] and natural language processing [14, 25, 52, 53].

Based on the feature spaces of the source and target domains, TL can be categorised as homogeneous or heterogeneous [45]. The former refers to scenarios where the source and target domains have the same set of features (attributes), resulting in a

Table 1: Characteristics of ER data sets with feature vectors rounded to two decimal places. *Ambiguous* are the feature vectors with both match (M) and non-match (N) class labels. We refer to *Same class* as the common feature vectors that have the same class label in both source and target domains, and *Diff(erent) class* as those common feature vectors that have different class labels in the two domains. For the Isle of Skye (IOS) and Kilmarnock (KIL) data sets, Bp-Bp and Bp-Dp refer to links between birth parents across two birth certificates and birth parents linked to death parents, respectively, as we describe in Section 5.

Feature Vectors of Domain A						Feature Vectors of Domain B					Common Feature Vectors			
Num. attributes	Name	Total num.	M	N	Ambiguous	Name	Total num.	M	N	Ambiguous	Total num.	Same class	Diff. class	Ambiguous
4	DBLP-ACM [13]	6,660	29.9%	66.5%	3.6%	DBLP-Scholar [13]	16,041	33.2%	66.6%	0.2%	2,159	98.7%	1.1%	0.2%
5	Million Songs [13]	27,544	33.2%	64.3%	2.5%	Musicbrainz [34]	91,143	14.3%	63.6%	22.1%	25,491	34.6%	1.0%	64.4%
8	IOS Bp-Dp [50]	115,986	19.0%	65.9%	15.0%	KIL Bp-Dp [50]	242,457	15.0%	65.4%	19.6%	81,019	19.0%	0.8%	80.2%
11	IOS Bp-Bp [50]	249,396	25.4%	64.0%	10.6%	KIL Bp-Bp [50]	406,038	28.2%	58.8%	13.1%	133,986	41.0%	0.1%	58.9%

common feature space. With heterogeneous TL, the target attributes are completely different from the source attributes thus resulting in different feature spaces. For ER, it is not clear if heterogeneous TL actually makes sense, quoting Jin et al. [26]: "Transferring knowledge between irrelevant domains (e.g., celebrities and products) does not produce meaningful outputs".

As we discuss further in Section 2, to the best of our knowledge all existing TL approaches for ER assume the homogeneous scenario because the contribution of different attributes towards match and non-match decisions can be different in the source and target domains. Our work therefore also focuses on homogeneous TL, which will help in situations where the same types of attributes are available and the same similarity functions can be applied. In practice we can, for example, use publicly available linked data sets as the source domain. While the homogeneous TL scenario seems straight-forward, as we discuss next, the specific characteristics of ER still pose several challenges for homogeneous TL.

First, the source and target domains can have different marginal probability distributions because their data are drawn from two different domains. This makes it possible—but not advisable—to apply the same supervised learner.

Another challenge in applying TL to ER lies in the feature vectors that possibly have different labels in the two domains, known as the difference in class conditional probabilities [45]. This difficulty becomes evident upon a closer look at the class conditional distributions of record pairs in two domains. Table 1 shows pairs of semantically related domains which have common features along with their match and non-match percentages, and statistics about the feature vectors in common to both domains. An important aspect is that even within a particular domain there exist feature vectors that have both match and non-match labels, as shown in the *Ambiguous* columns. This occurs due to the limited number of attributes available (first column in Table 1), and because often each record pair is manually labelled as a match or non-match independently from all other pairs when training data are generated [9].

The *Common Feature Vectors* columns in Table 1 highlight the difficulty of the TL task for ER, showing the considerable amount of feature vectors in two domains that have ambiguous or different class labels. For example, from the Musicbrainz [34] data set, for the feature vector [1.0, 0.0, 1.0, 0.9] (with attributes Title, Album, Artist, and Year compared using approximate string and numerical comparators [9]), we identified the following two record pairs where the first pair is a true match and the second a true non-match:

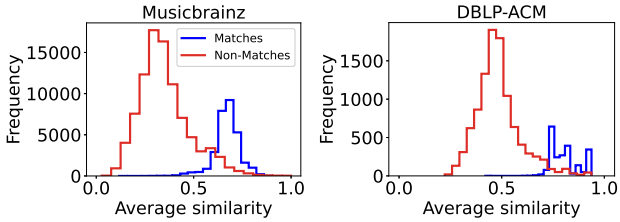

Figure 2: Skewed and bi-modal data distributions in the Musicbrainz [34] and DBLP-ACM [13] data sets.

[non e francesca, dies irae, formula 3, 1970]
[non e francesca, formula 3, formula 3, 1971]

[post modern sleaze, post modern sleaze ep, sneaker pimps, 1997]
[post modern sleaze, becoming remixed, sneaker pimps, 1996]

We do not assume perfect matching in either of the domains, rather both domains can have this kind of conflicting class labels.

Moreover, ER is generally a highly imbalanced classification problem with both skewed and bi-modal data distributions, as illustrated in Figure 2 on two data sets. For both of these, the distributions of similarities calculated between records have two peaks, where the peak with low average similarities accounts for non-matches (the vast majority of compared record pairs) and the high average similarities account for matches. As we experimentally show in Section 5, existing feature based TL methods [44, 52] fail on such problems since they are highly dependent on normally distributed data [52]. Furthermore, as is the case with most supervised ER problems, the low dimensionality of the feature space (similarities) generated when comparing records, as shown in the first column in Table 1, makes TL more difficult.

As we discuss further in Section 2, all existing TL approaches for ER use deep learning methods [7, 26, 28, 37, 54, 58]. These methods seem to be best suited for textual and unstructured data sets [1, 38] because of their ability to perform hierarchical representation learning [1]. Deep learning methods have recently seen an increased demand due to the requirements for matching lengthy textual data extracted from websites, such as descriptions of organisations, movies, consumer products, or tweets [6, 38].

In contrast, personal data such as health, census, or government records, are usually structured with only few attributes describing each entity [11]. These types of data are also known to contain typographical errors and spelling variations originating from manual data entry, scanning, or transcriptions [9]. Traditional machine learning methods, such as SVMs based on

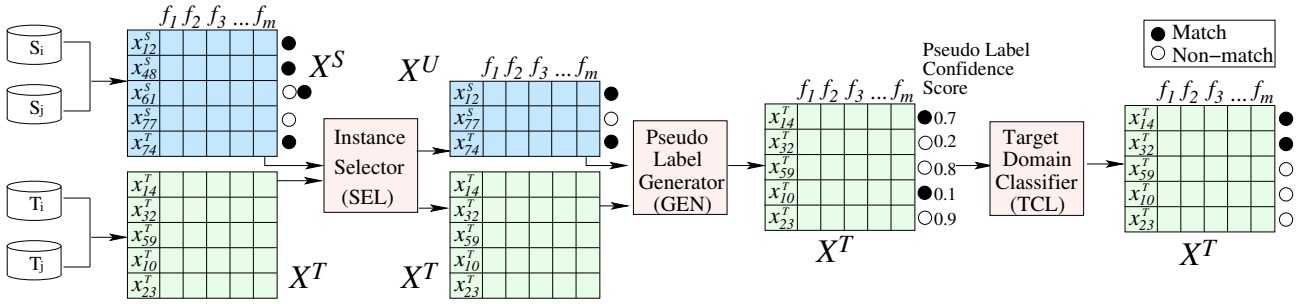

Figure 3: Overview of our *TransER* framework. As we describe in Section 4, given two databases, S_i and S_j, in the source domain, we first obtain candidate record pairs by blocking records [47]. Then we generate the feature matrix, X^S, where each feature, f_k represents an attribute similarity. In the same way, for the target domain databases, T_i and T_j, we obtain the feature matrix, X^T. Given X^S and X^T, the instance selector chooses source instances with a high class confidence and a similar local neighbourhood to the instances in the target to be transferred into X^U. Next, we generate pseudo labels for X^T from X^U with a score that provides the confidence of a pair being a match or non-match. Finally, X^T is classified using a classifier trained on the target instances that have high confidence pseudo labels. Note that source instance $x^S_{61} \in \mathrm{X}^S$ has conflicting class labels.

similarity measures, have shown to perform better on such data compared to deep learning models due to the small number of attributes available [1, 28, 38]. Furthermore, deep learning methods generally require a large number of labelled examples for training. Kasai et al. [28], for example, showed that for small data sets traditional machine learning based models have better performance than deep learning models.

To the best of our knowledge, we are the first to introduce a model agnostic TL method for ER that incorporates traditional machine learning models for structured data. As we show in the evaluation in Section 5, for such data sets existing TL methods for ER based on deep learning perform worse than those using features based on similarity functions.

As illustrated in Figure 3, our framework has three key phases: (1) an *instance selector (SEL)*, (2) a *pseudo label generator (GEN)*, and (3) a *target domain classifier (TCL)* that provides the final labels for the target instances. As we describe in Section 4, the instance selector ignores those source instances that have different class conditional probability distributions to similar target instances. The selected source instances, along with the pseudo labelling and target domain classifier, allow us to overcome issues arising from the differences in the marginal probability distributions between the source and target domains. Moreover, as we use the local neighbourhoods of instances in the transfer process rather than the whole data distributions as in existing feature based transfer methods, we minimise the influence of class imbalance. While our study contributes to TL for ER, it will also be of interest to general TL research, specifically for domains that have non-normal and imbalanced data distributions.

In particular, we make the following contributions: (1) We propose a novel TL framework for ER of structured data named *TransER*, (Transfer learning for ER). (2) We present three key phases in *TransER* that address the differences in marginal probability distributions and the class conditional probabilities in domains with non-normal data distributions. (3) We conduct extensive experiments in Section 5 to illustrate how *TransER* can be applied, and show that it can outperform several state-of-the-art methods.

2 RELATED WORK

We now review research related to the areas of ER and TL, and detail recent work that utilises TL in the context of ER.

Since the 1950s, ER techniques have been developed to link records across different databases [9, 17, 46]. Based on recent research, ER can be categorised into supervised, unsupervised, and semi-supervised approaches. Supervised approaches exist for learning similarity measures [4] and learning the classification task [16, 29, 43] through a model to predict the match and non-match labels of unlabelled instances (record pairs). Recently, deep learning models [6, 19, 38, 42] have been proposed that provide better results than traditional supervised methods, however at the cost of requiring large training data sets. Unsupervised approaches include traditional ER techniques [9] that depend on attribute similarities to decide on the class label of a record pair, as well as clustering [18, 24, 39] and collective ER approaches [3, 15, 27] that incorporate relational information for classifying a record pair. Semi-supervised ER techniques include active learning approaches [28, 49] that query external sources to resolve challenging training cases, as well as crowd based systems [21, 57] that employ hybrid machine and human based systems for resolving entities.

For TL, different solutions have been proposed [45], including *feature representation* and *instance* based transfer methods. Feature representation based methods [14, 44, 52] find a common latent feature space that minimises the difference in the distributions between the source and target domains. The dependence of these methods on normally distributed data [52] makes it challenging to apply them on ER, because ER is generally a highly imbalanced problem with both skewed and bi-modal data distributions (more non-matches than matches) [9], as can be seen in Figure 2.

Instance transfer methods can be categorised as instance reweighting and instance selection methods [45]. Instance reweighting [8, 12, 53] attempts to correct the differences in data distributions by reweighting the samples in the source domain to match the distribution in the target domain [45]. However, when the class conditional distribution is different across domains, instance selection [25, 35, 56] is more suitable as it removes instances with conflicting labels. To prune such instances with conflicts, Liao et

al. [35] utilised an active learning method while Jiang et al. [25] pruned instances in a different problem setting to ours where a small amount of target domain labels are available.

Vercruyssen et al. [56] selected instances that have similar local distributions in both source and target domains. While their *LocIT* method is based on a similar intuition to our work, it is different from ours in several aspects. First, it does not consider the local class label distribution in the source domain when selecting an instance. Second, their method uses a supervised classifier to select instances whereas we employ an unsupervised approach. Third, *LocIT* employs data distribution properties in the application domain of anomaly detection to select instances, while our framework is independent of such properties.

In the ER literature, only a few studies have focused on TL [7, 26, 28, 37, 41, 54, 58]. Explicitly focused on multi-source similarity learning in the record pair comparison step, Negahban et al. [41] incorporated TL to share learned structures between scoring problems. Zhao et al. [58], Chen et al. [7], Loster et al. [37], and Jin et al. [26] employed deep learning based TL models for ER to learn transferable attribute specific similarity functions for the record comparison step, and then used deep learning models in the classification step. These deep learning models have been shown to be better suited for long textual and unstructured data [1, 38] since the distributed feature representations used by these methods are most suitable to represent such textual data.

In the context of structured data, such as personal data [11], deep learning models, however, seem to perform worse than traditional machine learning methods that are based on features calculated on similarity measures, and when the available training data sets are small [1, 28]. Therefore, Kasai et al. [28] have proposed an active learning framework on top of deep transfer models to enhance the results for structured data. Although their method provided promising results, the proposed solution required manually labelled data in the target domain for the active learning step. Thirumuruganathan et al. [54] learned a transferable model using distributed feature representations for ER, where the authors have only provided results when partially labelled data are available in the source domain. However, as we show in our evaluation in Section 5, when the source domain is fully labelled, their method can result in poor performance.

To the best of our knowledge, all TL models proposed for ER are addressing the homogeneous scenario. For example, Loster et al. [37] perform a one-to-one mapping of attributes from source to target, while Zhao et al. [58] utilise pre-trained models of the same attribute type for transferring. In ER, researchers are interested in the homogeneous TL scenario because different attributes can have different contributions towards the match and non-match classifications in the source and target domains. The applicability of heterogeneous TL for ER has been questioned by Jin et al. [26].

3 PROBLEM DEFINITION

Let $\mathbf{R}$ be a set of records from a single or two databases, that we aim to link, and $\mathbf{B} \subset \mathbf{R} \times \mathbf{R}$ a subset of candidate record pairs obtained from blocking or indexing [9, 47] to avoid the quadratic record pair comparison space corresponding to all pairs in $\mathbf{R} \times \mathbf{R}$. Each record $r \in \mathbf{R}$ contains values, v_a, for a set of attributes, $a \in \mathbf{A}$, that provide information such as the name, address, and age of a person; or the author names, venue, and title for a publication.

Each candidate record pair $(r_i, r_j) \in \mathbf{B}$ is represented by an m-dimensional feature vector $x_{ij} \in \mathbb{R}^m$. A feature, f_q, with $1 \leq q \leq m$, represents the similarities calculated by comparing the values of an attribute, $a_q \in \mathbf{A}$. A feature value is calculated as $x_{ij}[q] = sim_a(r_i.v_{a_q}, r_j.v_{a_q})$ where $sim_a()$ is a similarity function appropriate to the type of values in a_q, such as the Jaro-Winkler similarity for names [9]. Let $\mathbf{X} \in \mathbb{R}^{n \times m}$ be the feature matrix generated by all feature vectors of candidate record pairs, such that $\mathbf{X} = \{x_{ij} : \forall (r_i, r_j) \in \mathbf{B}\}$ and $n = |\mathbf{B}|$ is the number of record pairs. Moreover, let $\mathbf{Y} \in \mathbb{R}^n$ be the vector of class labels of each record pair, such that $\mathbf{Y} = \{y_{ij} : \forall (r_i, r_j) \in \mathbf{B}\}$ with $|\mathbf{Y}| = n$ and $y_{ij} \in \{1,0\}$, where 1 and 0 represent a match and a non-match, respectively.

For the purpose of defining the problem of TL for ER, we first define a *domain* and a *task* [45]. A domain $\mathcal{D}$ consists of a feature space $\mathcal{X}$ of all record pairs and a marginal probability distribution $\mathcal{P}(\mathbf{X})$, where $\mathbf{X} \in \mathcal{X}$ is the feature matrix in a particular domain. A task $\mathcal{T}$ consists of a label space $\mathcal{Y}$, with $\mathbf{Y} \in \mathcal{Y}$, and a predictive function C that classifies instances and maps them to the corresponding class label. Following [45], the predictive function C can be rephrased as the class conditional probability distribution $\mathcal{P}(\mathbf{Y}|\mathbf{X})$ between the class labels and feature vectors from a probabilistic point of view. Following [45], let $\mathcal{D}^S$ and $\mathcal{T}^S$ be the source domain and task where $\mathcal{D}^S = \{\mathcal{X}^S, \mathcal{P}(\mathbf{X}^S)\}$ and $\mathcal{T}^S = \{\mathcal{Y}^S, \mathcal{P}(\mathbf{Y}^S|\mathbf{X}^S)\}$. Similarly, the target domain and task are $\mathcal{D}^T = \{\mathcal{X}^T, \mathcal{P}(\mathbf{X}^T)\}$ and $\mathcal{T}^T = \{\mathcal{Y}^T, \mathcal{P}(\mathbf{Y}^T|\mathbf{X}^T)\}$, respectively. Note that we use superscripts for defining the domain ('S' for source and 'T' for target) and subscripts for indexing of feature vectors and records.

Generally, in TL, the source and target databases can differ either with different domains, $\mathcal{D}^S \neq \mathcal{D}^T$, or with different tasks, $\mathcal{T}^S \neq \mathcal{T}^T$ [45]. The condition $\mathcal{D}^S \neq \mathcal{D}^T$ stands for either $\mathcal{X}^S \neq \mathcal{X}^T$ or $\mathcal{P}(\mathbf{X}^S) \neq \mathcal{P}(\mathbf{X}^T)$. In this paper, we consider the homogeneous case where the source and target domains have the same feature space $\mathcal{X}^S = \mathcal{X}^T$. This means the same similarity functions $sim_a()$ are used on attributes $a \in \mathbf{A}$ that contain the same type of information in both $\mathcal{D}^S$ and $\mathcal{D}^T$. However, as the data is drawn from two different domains we have $\mathcal{P}(\mathbf{X}^S) \neq \mathcal{P}(\mathbf{X}^T)$, which is referred to as the difference in *marginal probability distributions* [45].

Similarly, the condition $\mathcal{T}^S \neq \mathcal{T}^T$ stands for either $\mathcal{Y}^S \neq \mathcal{Y}^T$ or $\mathcal{P}(\mathbf{Y}^S|\mathbf{X}^S) \neq \mathcal{P}(\mathbf{Y}^T|\mathbf{X}^T)$. In terms of the task, all ER problems have the same label space of matches and non-matches, $\mathcal{Y}^S = \mathcal{Y}^T$, with $|\mathcal{Y}^S| = |\mathcal{Y}^T| = 2$. However, as the same feature vector can have different labels in the two domains we have $\mathcal{P}(\mathbf{Y}^S|\mathbf{X}^S) \neq \mathcal{P}(\mathbf{Y}^T|\mathbf{X}^T)$, which is referred to as the difference in *class conditional distributions* [45] and can be seen in Table 1 (the *Ambiguous* columns). We now formally define the problem of TL in the ER setting.

Definition 3.1. **Transfer learning for entity resolution**: Consider we have a source domain $\mathcal{D}^S$ and a task $\mathcal{T}^S$ with the feature matrix of records pairs $\mathbf{X}^S = \{x_{ij}^S : \forall (r_i, r_j) \in \mathbf{B}^S\}$ and their corresponding match and non-match labels $\mathbf{Y}^S = \{y_{ij}^S : \forall (r_i, r_j) \in \mathbf{B}^S\}$, and a target domain $\mathcal{D}^T$ with the target feature matrix $\mathbf{X}^T = \{x_{ij}^T : \forall (r_i, r_j) \in \mathbf{B}^T\}$. We assume both the source and target domains have the same feature space $\mathcal{X}^S = \mathcal{X}^T$ and the same label space $\mathcal{Y}^S = \mathcal{Y}^T$, but they can have different marginal probability distributions, $\mathcal{P}(\mathbf{X}^S) \neq \mathcal{P}(\mathbf{X}^T)$, and different class conditional distributions, $\mathcal{P}(\mathbf{Y}^S|\mathbf{X}^S) \neq \mathcal{P}(\mathbf{Y}^T|\mathbf{X}^T)$. The transfer learning problem for entity resolution is to predict the match and non-match labels for the record pairs in the target domain $\mathbf{Y}^T = \{y_{ij}^T : \forall (r_i, r_j) \in \mathbf{B}^T\}$ using $\mathcal{D}^S$, $\mathcal{T}^S$ and $\mathcal{D}^T$.

Algorithm 1: ***TransER***

Input: - $\mathbf{X}^S$: Feature matrix of source domain
- $\mathbf{Y}^S$: Labels of source domain
- $\mathbf{X}^T$: Feature matrix of target domain
- C: Classification method
- k: Neighbourhood size
- b: Class imbalance ratio
- t_c: Threshold for instance confidence similarity
- t_l: Threshold for instance structural similarity
- t_p: Threshold for pseudo label confidence

Output - $\mathbf{Y}^T$: Labels of target domain instances

// **Phase (i)**: Instance Selector (*SEL*)
$\mathbf{X}^U = \{\}$ // Initialise $\mathbf{X}^U$ to hold transferred instances
for $x^S \in \mathbf{X}^S$ **do:** // Iterate over source instances
  $\mathbf{N}_x^S = GetNeighbourhood(x^S, k, \mathbf{X}^S)$ // In source domain
  $\mathbf{N}_x^T = GetNeighbourhood(x^S, k, \mathbf{X}^T)$ // In target domain
  $sim_c = GetInstanceConfidenceSimilarity(x^S, \mathbf{N}_x^S)$
  $sim_l = GetInstanceStructuralSimilarity(x^S, \mathbf{N}_x^S, \mathbf{N}_x^T)$
  if $sim_c \geq t_c$ **and** $sim_l \geq t_l$ **then**:
    $\mathbf{X}^U = \mathbf{X}^U \cup \{x^S\}$ // Add x^S to the set of transferred instances
$\mathbf{Y}^U = GetLabels(\mathbf{X}^U, \mathbf{Y}^S)$ // Get labels of transferred instances
// **Phase (ii)**: Pseudo Label Generator (*GEN*)
$C^U = TrainClassifier(C, \mathbf{X}^U, \mathbf{Y}^U)$ // Train on transferred instances
$\mathbf{Y}^P, \mathbf{Z}^P = C^U.predict(\mathbf{X}^T)$ // Predict pseudo label confidence score
// **Phase (iii)**: Target Domain Classifier (*TCL*)
$\mathbf{X}^V = \{\}$ // Initialise to hold instances with high pseudo label confidence
for $x^T \in \mathbf{X}^T$ **do:** // Iterate to find target instances with high confidence
  $z^T = GetPseudoLabelConfidenceScore(x^T, \mathbf{Z}^P)$
  if $z^T \geq t_p$ **then**:
    $\mathbf{X}^V = \mathbf{X}^V \cup \{x^T\}$ // Add x^T to the set of candidate target instances
$\mathbf{Y}^V = GetLabels(\mathbf{X}^V, \mathbf{Y}^P)$ // Get pseudo labels of candidate instances
$\mathbf{X}_b^V, \mathbf{Y}_b^V = GetBalancedData(\mathbf{X}^V, \mathbf{Y}^V, b)$
$C^V = TrainClassifier(C, \mathbf{X}_b^V, \mathbf{Y}_b^V)$ // Train on balanced instances
$\mathbf{Y}^T = C^V.predict(\mathbf{X}^T)$ // Predict all final target labels
return $\mathbf{Y}^T$

4 INSTANCE BASED TRANSFER LEARNING

We now describe our framework *TransER* that employs instance based TL for ER on homogeneous and structured data. As discussed in Section 1 and illustrated in Figure 1, the ER process includes the steps of blocking, record pair comparison, classification, and evaluation [9]. For all steps in this process, we use standard techniques as for example discussed in [9, 17, 46].

We contribute to the classification step to reduce the labelling costs for supervised learning approaches by reusing already labelled data available in a semantically related domain. We classify record pairs as matches or non-matches utilising traditional machine learning methods and features calculated using similarity measures [9]. We prefer traditional machine learning models over deep learning approaches because the former seem to provide competing or better results and higher efficiency for structured data, as shown by Mudgal et al. [38] and validated in our experiments in Section 5.

TransER consists of three key phases as illustrated in Figure 3: instance selector (*SEL*), pseudo label generator (*GEN*), and target domain classifier (*TCL*). Algorithm 1 details the phases of our framework, which we now describe in detail. Our framework can be applied for both resolving entities within a single or between two databases because all we require are the similarities calculated between records (in the form of the two feature matrices $\mathbf{X}^S$ and $\mathbf{X}^T$) and the labels in the source domain, $\mathbf{Y}^S$.

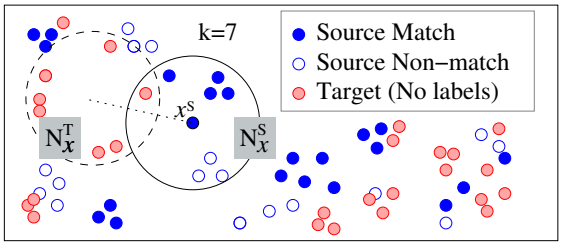

Figure 4: Source and target instances in a 2D feature space. In this example $k = 7$ and the instance x^S is a true match in the source, where $\mathbf{N}_x^S$ and $\mathbf{N}_x^T$ are the neighbourhoods of x^S in the source and target domains, respectively (notice how $\mathbf{N}_x^T$ are the 7 closest target instances in $\mathbf{X}^T$ to x^S). In the source domain (blue instances) the nearest neighbours of x^S consist of 4 matches and 3 non-matches. Therefore, based on Equation (1), $sim_c(x^S) = 4/7$. To calculate $sim_l(x^S)$ using Equation (2), the distance between $\mathbf{N}_x^S$ and $\mathbf{N}_x^T$ should be calculated, as indicated by the dashed line.

4.1 Instance Selector (*SEL*)

If a source instance $x^S \in \mathbf{X}^S$ has a different class label in the target domain, it contributes to the class conditional distribution difference we discussed in Section 3. As a solution to this problem, in the instance selector phase our aim is to select source instances that have the same representation (have the same class label) in the target domain. However, as we do not have any class labels in the target domain, we make two intuitive assumptions.

The first is based on the smoothness assumption in semi-supervised learning which states that if two instances are close, their class labels are likely the same [55]. For a given source instance, we assume it has a similar representation in the target domain only if the source instance has a high class label confidence based on its neighbouring instances also having the same class label. This is an important assumption for ER, because ER data sets consist of feature vectors that can have conflicting labels even within the same domain, as we illustrated in Table 1.

Second, inspired by the *LocIT* method [56], we assume that an instance has a similar representation in both the source and target domains if the local structure (neighbouring feature vectors) of the marginal distributions around the instance are similar in both domains. The use of local structures becomes beneficial in the context of ER due to the bi-modal data distributions generally seen in ER data sets (as shown in Figure 2) which can make it difficult to apply most existing statistical models for feature transformation on the full data distributions [44, 52].

Let k be the neighbourhood size of an instance as its k closest neighbours. In order to characterise the class confidence and local structure of an instance $x^S \in \mathbf{X}^S$, we first define the *local source distribution* of an instance as the neighbourhood $\mathbf{N}_x^S$ of x^S in $\mathbf{X}^S$ limited to its k nearest neighbours. Similarly, we define the *local target distribution* as the neighbourhood $\mathbf{N}_x^T$ of x^S in $\mathbf{X}^T$ limited to its k nearest neighbours in $\mathbf{X}^T$. We have $|\mathbf{N}_x^S| = |\mathbf{N}_x^T| = k$.

As illustrated in Figure 4 and based on the two assumptions stated above, we now propose two similarity functions to characterise an instance for transferability.

The first similarity function represents the class label confidence of an instance. If $|\{x_\alpha : x_\alpha \in \mathbf{N}_x^S \wedge y_\alpha = y^S\}|$ is the number

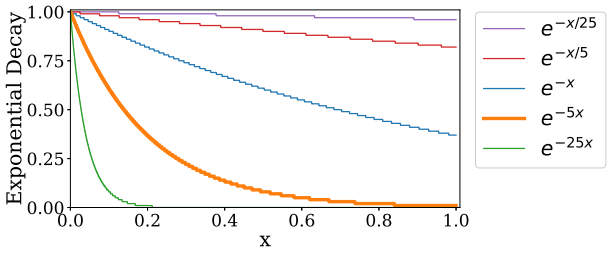

Figure 5: Behaviour of exponential decay functions.

of neighbours in $\mathbf{N}_x^S$ with the same class label as the source instance x^S, then we characterise the class confidence of x^S as follows:

$$sim_c(x^S) = |\{x_\alpha : x_\alpha \in \mathbf{N}_x^S \wedge y_\alpha = y^S\}| \; / \; k, \tag{1}$$

Based on our second assumption for x^S to be transferable, the distribution of $\mathbf{N}_x^S$ should be similar to $\mathbf{N}_x^T$. To measure this structural similarity of an instance, we utilise the location distance used in *LocIT* [56], which considers the L2-norm difference between the centroids of the two neighbourhood sets $\mathbf{N}_x^S$ and $\mathbf{N}_x^T$ as the location distance. To normalise this distance, we divide it by the maximum possible distance which is $\sqrt{m}$ because all our features are between 0 and 1, and where m is the number of features in the feature space.

Since we divide by the maximum possible distance, our normalised distance is more biased towards lower values. We therefore use an exponential decay function to convert this normalised distance value into a similarity score. Based on a set of initial experiments, we found that using $e^{-5(\cdot)}$ as the decay function provided us with values that are in the desired 0 and 1 interval, as illustrated in Figure 5. We define our second similarity function as follows:

$$sim_l(x^S) = e^{-5\left(\left\| \frac{1}{k}\left(\sum_{x_\alpha \in \mathbf{N}_x^S} x_\alpha - \sum_{x_\beta \in \mathbf{N}_x^T} x_\beta\right)\right\|_2 \Big/ \sqrt{m}\right)}, \tag{2}$$

If both the class confidence similarity and the structural similarity of a source instance x^S are at least the two thresholds, t_c and t_l, as set by the user (i.e. $sim_c(x^S) \geq t_c$ and $sim_l(x^S) \geq t_l$), then we consider x^S as a transferable instance.

Using $sim_c()$ and $sim_l()$, lines 1 to 9 in Algorithm 1 detail how we select source instances to transfer in the *SEL* phase. In line 1 we first initialise $\mathbf{X}^U$ as an empty set that we use to hold the instances selected for transferring. Then, in lines 2 to 8, the algorithm iterates through each source instance $x^S \in \mathbf{X}^S$, obtains its source and target neighbourhoods $\mathbf{N}_x^S$ and $\mathbf{N}_x^T$, and then calculates sim_c and sim_l as per Equations (1) and (2). If sim_c is at least the threshold t_c and sim_l is at least the threshold t_l, then we select x^S to be transferred and add it to $\mathbf{X}^U$. Finally, we obtain the labels of the selected instances as $\mathbf{Y}^U$ in line 9 using $\mathbf{X}^U$ and $\mathbf{Y}^S$. In Section 5.3 we discuss how selecting these thresholds affects our TL framework.

The time complexity of the nearest neighbour generation depends on the method employed. Assuming we use the KD tree algorithm [2], obtaining nearest neighbours for source and target has a time complexity of $O(m \cdot |\mathbf{X}^S| \cdot \log(|\mathbf{X}^S|) + m \cdot |\mathbf{X}^T| \cdot \log(|\mathbf{X}^T|))$ where m is the number of features. As nearest neighbour querying has a complexity of $O(log(|\mathbf{X}|))$ [2], the *SEL* phase has a time complexity of $O(|\mathbf{X}^S| \log(|\mathbf{X}^S|)) + O(|\mathbf{X}^S| \log(|\mathbf{X}^T|))$.

4.2 Pseudo Label Generator (*GEN*)

After selecting instances for transferring, we have $\mathbf{X}^U$ containing all instances that have a similar class conditional distribution to the target domain as per our assumptions. If we use $\mathbf{X}^U$ to predict the labels of $\mathbf{X}^T$, then we might not capture the marginal probability distribution in $\mathcal{D}^T$. Therefore, in this phase we generate pseudo labels for the target instances $x^T \in \mathbf{X}^T$. Then, in the next phase, we use these pseudo labels to train a classifier on the target domain itself. This way we can incorporate the marginal probability distribution of $\mathcal{D}^T$ into the classification.

To generate such pseudo labels, we first train a classifier C on the selected instances, $\mathbf{X}^U$, and their labels, $\mathbf{Y}^U$, in line 10 of Algorithm 1. We assume the classifier returns the probability of the predicted label for each instance that indicates the likelihood for an instance to belong to the predicted class. We use these probabilities as the pseudo label confidence score to select candidate target instances to train the final classifier. Therefore, we apply the trained classifier C^U on $\mathbf{X}^T$ to predict the pseudo labels $\mathbf{Y}^P$ along with a confidence score for each label $\mathbf{Z}^P$ in line 11 of the algorithm.

Assuming the worst case time complexity of the classifier, C, is $O(\Phi(\cdot))$, the time complexity of this second phase is $O(\Phi(|\mathbf{X}^U|))$.

4.3 Target Domain Classifier (*TCL*)

Now we have pseudo labels $\mathbf{Y}^P$ generated for each target instance $x^T \in \mathbf{X}^T$ along with their confidence score $\mathbf{Z}^P$. In this final phase, we aim to choose target instances that have a high confidence score of their pseudo labels and use them to train a classifier on the target. This process addresses two challenges we discussed in Section 1. First, we address the issue of the differences in marginal probability distributions because we use pseudo labels and train the final classifier on the target domain itself.

Second, we address the problem of class imbalance by balancing the classes of pseudo labels in the target domain. In ER data sets, most often the number of non-matches is much larger (sometimes in magnitudes) than the number of matches [9]. As we do not have labels for any instances in the target domain, the blocked record pairs are also prone to this class imbalance. Therefore, as a solution, we balance the target instances in $\mathbf{X}^T$. For this, we use the pseudo labels obtained from the previous phase and under-sample the non-match instances such that the ratio between matches and non-matches is at a predefined class imbalance ratio b. We set b based on a value commonly used in the ER literature [29, 38, 54], as we further discuss in Section 5.

In Algorithm 1, line 12 initialises $\mathbf{X}^V$ to an empty set to hold the candidate instances that have a high pseudo label confidence score. Then in lines 12 to 16, we iterate through each target instance $x^T \in \mathbf{X}^T$. If the confidence score of an instance's pseudo label z^T is greater than a predefined threshold t_p, then we select the instance and add it to $\mathbf{X}^V$. We then obtain the corresponding labels of the selected candidate instances in line 17. We under-sample non-matches to obtain a ratio b of non-matches to matches in line 18, and get $\mathbf{X}_b^V$ and $\mathbf{Y}_b^V$. Finally, in lines 19 and 20, we train a classifier, C^V, on the balanced sample with pseudo labels and apply it on the target data set $\mathbf{X}^T$ to obtain the final target instance labels, $\mathbf{Y}^T$.

Assuming the worst case time complexity of the classifier, C, is $O(\Phi(\cdot))$, the time complexity of this phase is $O(|\mathbf{X}^T| + \Phi(|\mathbf{X}_b^V|))$.

5 EXPERIMENTAL EVALUATION

We conduct an extensive set of experiments to address the following questions: (1) How does *TransER* perform compared to other state-of-the-art TL methods, both generic methods as well as those specifically aimed at ER? (2) How does the percentage of existing labelled source instances affect linkage quality? (3) How do the different parameters of *TransER* affect linkage quality? (4) How does each key component in *TransER* affect linkage quality?

5.1 Experimental Setup

5.1.1 Settings. For the blocking step, we employ a locality sensitive hashing based technique that maps records with similar attribute pairs to the same minhash value to group potential matches [47]. In the record pair comparison step we then employ similarity functions such as the Jaro-Winkler similarity for names and the Jaccard similarity for other textual strings [9] to compare attribute values between records. Then, we use the two feature matrices, $\mathbf{X}^S$ and $\mathbf{X}^T$, generated after the record comparison step. After transferring is conducted, we use a set of classifiers [48] including a support vector machine, a random forest, a logistic regression, and a decision tree, for classification, and we average their linkage quality results.

We implemented *TransER* and all baselines (using the transfertools[1] and deep matcher[2] libraries provided by [38, 56]) in Python 3.6 and using Sklearn 0.21 [48]. All experiments were conducted on a server running Ubuntu 18.04 with 64-bit Intel Xeon 2.10 GHz CPUs, and 512 GBytes of memory. We restricted the memory consumption of each experiment to 200 GBytes of main memory, and set the maximum runtime of an experiment to 72 hours. The code and feature matrices are available in an online repository[3].

We set the default *TransER* parameter values used for all experiments as $t_c = 0.9$, $t_l = 0.9$, $t_p = 0.99$, and $k = 7$ based on the parameter sensitivity analysis we conduct in Section 5.3, where we set the class imbalance b to have a match to non-match ratio of $1:3$ based on the ratios used in existing ER frameworks [29, 38, 54].

5.1.2 Data sets. We evaluate our framework on seven data sets from different domains, as detailed in Table 1. These include publicly available data sets [13, 34] from the bibliographic (ACM, DBLP, and Scholar [13]) and music (Million Songs (MSD) [13] and Musicbrainz (MB) [34]) domains, as well as proprietary data sets from the demographic domain, where for these data sets the goal is to link person records across birth, marriage, and death certificates [10, 39, 50]. To evaluate *TransER* and show the generalisability of our framework, we selected data sets with different characteristics, different sizes, and different levels of difficulty to match records.

Given DBLP-ACM consists of links between two well-structured data sets, it can be considered as a simple scenario to resolve [32]. However, DBLP-Scholar is more challenging because publications in Scholar have various data quality problems, such as misspellings and different representations of authors and venues [32]. These bibliographic data sets are fairly small generating only a few thousand feature vectors, as can be seen in Table 1. We therefore also used the music data sets that have resulted in more feature vectors.

[1]https://github.com/Vincent-Vercruyssen/transfertools
[2]https://github.com/anhaidgroup/deepmatcher
[3]https://github.com/nishadi/TransER

The demographic data sets consist of two data sets from Scotland [50], one from the remote Isle of Skye (IOS) and the other from the town of Kilmarnock (KIL). Both contain civil certificates (birth, marriage, and death) of their population over the period from 1860 to 1901. Demographers with expertise in linking such data have curated and manually linked certain types of relationships in these data sets, such as *Bp-Bp* (links between birth parents across two birth certificates) and *Bp-Dp* (birth parents linked to death parents), and we use those in our experiments. These data sets are much larger compared to the music and bibliographic data sets, and are challenging to resolve as they contain many data quality problems such as spelling errors and variations [50].

We paired data sets as shown in Table 1 based on the existence of common feature spaces. We show a source and target data set pair as "*source* → *target*" for the remainder of this section.

5.1.3 Baselines. As baselines we selected a set of methods that cover a wide range of related work, including TL models specifically for ER, as well as TL models that address the key challenges of different marginal probability distributions, different class conditional distributions, and class imbalance, as we discussed in Section 2.

- *Naive* is a baseline that blindly applies a classifier trained on the source domain, $\mathcal{D}^S$, to the target domain, $\mathcal{D}^T$ (this means no TL is conducted). This is similar to state-of-the-art baselines such as Magellan [29] and Tamer [51], where we can generate features based on similarity metrics and then we can choose a classifier such as a random forest, SVM, or logistic regression for the classification of record pairs.
- *DTAL** is a variant of the method proposed by Kasai et al. [28] which has achieved high performance with deep transfer models for ER in low resource settings. In their original proposal, the authors used both TL and active learning to improve results. Since the active learning approach requires labelled data on the target domain, for a fair comparison we consider *DTAL** without including the active learning aspect. Transferring is conducted by introducing a gradient reversal layer to a deep learning model. In our experiments, this method is a representative of deep TL models for ER.
- *DR* is a baseline proposed by Thirumuruganathan et al. [54] that uses deep learning with FastText [5] embeddings for feature representation and traditional machine learning models for classification. We average the results of a SVM, a random forest, a logistic regression, and a decision tree classifier. Transferring is conducted using instance weighting methods.
- *LocIT** is a variant of the TL method proposed by Vercruyssen et al. [56]. The original *LocIT* method has two parts, where in the first it selects instances for transferring and in the second it performs anomaly detection. In our context, we only employed *LocIT** for the instance selection part and then train an ER classifier on the selected instances. This baseline is the closest to *TransER* as it also uses the local structure of marginal distributions around the instances for instance selection. Therefore, this method is a representative of instance selection transfer models in our experiments.
- *TCA* [44] and *Coral* [52] are two state-of-the-art feature based TL baselines that transform the source and target

domains into a common subspace. Even though both methods have been successfully used for general TL, in our evaluation we show why they are not suitable methods for ER.

As we discussed in Section 2, existing deep learning methods for ER [7, 26, 28, 37, 58] are based on the embedding of attribute values into high-dimensional vectors followed by learning of similarity scores based on these embeddings. In the survey by Barlaug and Gulla [1], *DTAL* [28] showed comparative and even better results compared to most other deep learning methods evaluated on public benchmark data provided by Mudgal et al. [38]. We therefore use *DTAL* as a state-of-the-art approach to represent the class of deep learning TL methods. Furthermore, deep learning methods such as *Auto-EM* [58] are highly dependent on large knowledge bases and they do not work on small data sets [28] such as the ones we use with *TransER*. If we were to run methods such as *Auto-EM* on such small data sets, their architecture would deviate significantly from their original proposal [58].

5.1.4 Linkage Quality Evaluation Measures. To assess the linkage quality of *TransER* and the baselines, we compare four evaluation measures: precision (P), recall (R), the *F1*-measure, and the F^*-measure [23]. To describe what each evaluation measure represents, we consider TP, FP, and FN as the number of true matches, false matches, and false non-matches, respectively [9]. Precision is the number of true matches against the total number of matches generated by a particular method, $P = TP/(TP+FP)$; and recall is the number of true matches against the total number of matches in the linked ground truth data, $R = TP/(TP + FN)$ [22].

To facilitate a comparison of results, it is best to use a single number measure that captures aspects of both precision and recall [9, 23]. Recent research has found that the *F1*-measure is not suitable for measuring ER quality [22] because the relative importance given to precision and recall in this measure depends on the number of predicted matches. We therefore use the alternative, more interpretable, F^*-measure [23] calculated as $F^* = TP/(TP + FP + FN)$. The F^* measure corresponds to the number of true matches against the number of matches which are either misclassified or are correctly classified true matches. We also show the *F1*-measure to facilitate comparisons with other published results.

5.2 TransER Performance

We now compare the performance of *TransER* with the aforementioned baselines with respect to linkage quality and runtime. We then show how scalable the *TransER* framework is with regard to the percentage of labels available in the source domain.

5.2.1 Linkage Quality Comparison. Table 2 provides the precision, recall, F^*, and *F1* results of *TransER* and of the baselines evaluated. Based on the average results (lower part of the table), we can see that *TransER* outperforms all baselines while most of the state-of-the-art TL methods are not able to consistently outperform the *Naive* baseline.

It is important to discuss the reason behind poor ER quality results of the TL baselines in comparison to the *Naive* baseline. As we discussed in Section 1, we are the first to utilise traditional machine learning models with similarity measure based feature representations in the context of TL for ER. Therefore, our *Naive* baseline is also based on those traditional machine learning models. Existing deep learning models for ER work better for long textual data and they seem to be less successful with structured data [38]. This is the reason for the poor performance of deep transfer ER baselines such as *DTAL**. Although *LocIT** employs a similar instance selection model as *TransER*, it also cannot outperform the *Naive* baseline because *LocIT** is optimised for anomaly detection. We can also see that *TCA* outperforms the *Naive* baseline only on the simple and small DBLP-ACM → DBLP-Scholar data set pair. Both *TCA* and *Coral* perform feature representation based TL and therefore are dependent on normally distributed data, whereas ER generally has bi-modal and skewed data distributions. This results in a poor performance of these baselines.

TransER outperforms the *Naive* baseline on all of the data set pairs (except for DBLP-Scholar → DBLP-ACM) with an average improvement of 10% of F^* and 9% of *F1*. It is important to note that the recall is improved for up-to 50% because *TransER* addresses the issues of differences in marginal probability distributions and conditional probability distributions. However, we can see that *TransER* and the *Naive* baseline have competing results for DBLP-Scholar → DBLP-ACM because DBLP-ACM is a fairly small and simple data set to resolve [32]. From Table 1 we can also see that the DBLP-Scholar data set has well separated match and non-match feature vectors with only 0.2% ambiguous feature vectors. Therefore, the *Naive* baseline itself can achieve good results for this scenario, and the instance selection we use in *TransER* has a only little impact. Other than with this data set pair, for all other (more challenging) pairs a significant improvement in the results for *TransER* can be seen, with up-to 36% for F^* and 41% for the *F1*-measure.

*DTAL** is a representative of deep TL models proposed for ER. We report results of only the small and middle sized data sets due to the very long runtimes taken for training these deep learning models for the larger data sets (see Table 3). However, as we can see from Table 2, *DTAL** is not achieving acceptable results when compared with both *TransER* and the *Naive* baseline. These results align with experiments presented previously [1, 38] for structured data. As pointed out by Mudgal et al. [38], deep learning models seem to be better suited for unstructured textual data consisting of long values that contain multiple words, because these models capture semantic similarities from such textual data.

However, in our work we concentrate on structured data with short strings that can contain typographical errors and spelling variations (as commonly occur with personal data [9]). We can see that deep learning models are less successful in representing such features compared to the (string) similarity based feature representations. Although *DTAL** achieves comparative results for simpler scenarios such as DBLP-Scholar → DBLP-ACM, we can see that its results drop drastically for more challenging scenarios such as MSD → MB, with a drop of 44% in F* and 57% in *F1*.

DR is a baseline that uses deep learning models for feature representation and traditional machine learning for classification. As we can see in Table 2, it is one of the worst performing baselines. This poor performance compared also to the *Naive* baseline is referred to as *negative transfer* [54] by the authors. This is because the feature embeddings in *DR* are directly calculated from existing FastText embeddings and our data sets have many out-of-vocabulary words, such as person names and addresses. We do not have a rich context to approximate these out-of-vocabulary words with the limited number of attribute values in structured data.

Table 2: Precision (*P*), Recall (*R*), F^*, and *F1* results of *TransER* compared to the baselines. The results are presented as averages ± standard deviations for the experiments conducted over a set of classifiers, as described in Section 5. 'ME' and 'TE' refer to experiments which exceeded the limitations of 200 GBytes of memory or 72 hours of runtimes, respectively.

Source	Target		*TransER*	Naive	DTAL* [28]	DR [54]	LocIT* [56]	TCA [44]	Coral [52]
		P	92.78 ± 5.13	99.02 ± 0.21	92.74 ± 3.34	56.30 ± 2.74	0.00 ± 0.00	94.08 ± 6.06	96.30 ± 2.24
DBLP-	DBLP-	*R*	96.90 ± 1.27	83.41 ± 2.35	66.18 ± 10.50	61.37 ± 2.53	0.00 ± 0.00	67.65 ± 22.17	86.72 ± 11.79
ACM	Scholar	F^*	**90.02** ± 4.31	82.73 ± 2.35	62.64 ± 9.11	41.45 ± 0.36	0.00 ± 0.00	65.89 ± 22.24	83.60 ± 10.22
		F1	**94.69** ± 2.46	90.53 ± 1.40	76.65 ± 6.77	58.61 ± 0.37	0.00 ± 0.00	76.87 ± 19.26	90.72 ± 6.29
		P	93.76 ± 1.01	94.26 ± 1.91	92.00 ± 2.44	19.19 ± 5.23	14.98 ± 33.52	93.20 ± 0.74	90.75 ± 18.16
DBLP-	DBLP-	*R*	99.18 ± 0.43	99.46 ± 0.39	98.42 ± 0.18	3.84 ± 1.39	10.45 ± 27.69	99.26 ± 0.30	87.33 ± 25.90
Scholar	ACM	F^*	93.03 ± 0.72	**93.77** ± 1.67	90.67 ± 2.52	3.12 ± 0.85	9.79 ± 25.80	92.56 ± 0.92	84.07 ± 24.84
		F1	96.39 ± 0.39	**96.78** ± 0.89	95.09 ± 1.39	6.05 ± 1.61	11.35 ± 27.81	96.13 ± 0.50	88.07 ± 24.42
		P	73.18 ± 14.06	84.58 ± 4.53	23.40 ± 24.94	43.00 ± 0.29	TE	ME	86.50 ± 2.54
MSD	MB	*R*	62.04 ± 25.41	12.46 ± 2.15	3.47 ± 4.18	40.02 ± 19.42	TE	ME	69.68 ± 6.95
		F^*	47.83 ± 16.11	12.18 ± 2.07	3.25 ± 3.92	24.77 ± 8.10	TE	ME	**62.81** ± 6.16
		F1	63.09 ± 14.91	21.65 ± 3.28	6.02 ± 7.18	39.02 ± 10.47	TE	ME	**76.98** ± 4.89
		P	92.99 ± 2.19	90.06 ± 2.10	34.30 ± 1.31	21.59 ± 10.82	TE	ME	90.45 ± 9.18
MB	MSD	*R*	97.15 ± 2.79	93.54 ± 4.07	29.23 ± 3.41	25.20 ± 27.81	TE	ME	70.23 ± 18.93
		F^*	**90.47** ± 2.46	84.72 ± 3.19	18.65 ± 1.31	12.83 ± 12.42	TE	ME	64.53 ± 17.32
		F1	**94.98** ± 1.38	91.69 ± 1.90	31.41 ± 1.88	20.71 ± 18.49	TE	ME	76.93 ± 14.52
		P	95.98 ± 0.30	95.82 ± 0.38	TE	27.93 ± 0.90	TE	ME	92.24 ± 4.53
IOS-Bp-Dp	KIL-Bp-Dp	*R*	96.13 ± 0.81	95.02 ± 1.04	TE	2.19 ± 0.98	TE	ME	72.52 ± 17.34
		F^*	**92.41** ± 0.53	91.24 ± 1.17	TE	2.05 ± 0.88	TE	ME	68.41 ± 15.93
		F1	**96.05** ± 0.28	95.42 ± 0.65	TE	4.01 ± 1.69	TE	ME	80.16 ± 11.50
		P	91.97 ± 1.62	79.35 ± 12.91	TE	29.09 ± 1.63	TE	ME	77.61 ± 21.52
KIL-Bp-Dp	IOS-Bp-Dp	*R*	96.96 ± 0.41	96.59 ± 1.49	TE	1.66 ± 0.42	TE	ME	88.14 ± 8.75
		F^*	**89.39** ± 1.50	76.97 ± 11.89	TE	1.59 ± 0.38	TE	ME	70.98 ± 19.68
		F1	**94.39** ± 0.83	86.44 ± 8.18	TE	3.12 ± 0.74	TE	ME	81.20 ± 15.94
		P	80.34 ± 0.25	80.46 ± 0.25	TE	31.67 ± 1.29	TE	ME	77.47 ± 1.70
IOS-Bp-Bp	KIL-Bp-Bp	*R*	97.87 ± 0.23	96.26 ± 1.24	TE	2.98 ± 0.75	TE	ME	80.40 ± 9.66
		F^*	**78.96** ± 0.37	78.02 ± 0.92	TE	2.78 ± 0.65	TE	ME	65.18 ± 7.27
		F1	**88.25** ± 0.23	87.65 ± 0.58	TE	5.41 ± 1.24	TE	ME	78.68 ± 5.40
		P	88.08 ± 1.50	81.39 ± 14.12	TE	30.51 ± 0.21	TE	ME	73.05 ± 26.27
KIL-Bp-Bp	IOS-Bp-Bp	*R*	97.78 ± 0.29	83.99 ± 2.40	TE	1.64 ± 0.42	TE	ME	64.33 ± 23.93
		F^*	**86.35** ± 1.35	69.77 ± 9.85	TE	1.58 ± 0.40	TE	ME	50.11 ± 20.21
		F1	**92.67** ± 0.78	81.77 ± 7.27	TE	3.11 ± 0.77	TE	ME	64.10 ± 19.99
		P	88.64 ± 8.88	88.12 ± 9.57	60.61 ± 34.93	32.41 ± 12.06	7.49 ± 23.36	93.64 ± 4.02	85.55 ± 14.80
Averages		*R*	93.00 ± 14.34	82.59 ± 27.57	49.33 ± 37.61	17.36 ± 24.09	5.22 ± 18.97	83.46 ± 22.27	77.42 ± 17.27
		F^*	**83.56** ± 15.30	73.67 ± 25.25	43.80 ± 36.18	11.27 ± 14.69	4.89 ± 17.68	79.22 ± 20.39	68.71 ± 17.94
		F1	**90.06** ± 11.65	81.49 ± 23.72	52.29 ± 36.80	17.51 ± 20.89	5.67 ± 19.19	86.50 ± 16.28	79.61 ± 14.89

The *LocIT** baseline is the closest to *TransER* as it also uses the local structures of instances to decide on their transferability. However, as we can see in Table 2, *LocIT** is the worst performing baseline. Since it is proposed in the context of anomaly detection, it assumes two instances are not transferable if they are located far away. As this does not hold in the ER context, in some scenarios *LocIT** achieves 0% for all evaluation measures. Even if there exist instances that do adhere to its assumption, for example in the DBLP-Scholar → DBLP-ACM scenario, it still only achieves very low quality results.

Since the *TCA* baseline exceeded the available memory even for mid-sized data sets, we could conduct experiments only on the smallest data sets. Although *TCA* achieves competing results for the easy scenario DBLP-Scholar → DBLP-ACM, we observe that F^* drops by 24% when resolving the more challenging scenario DBLP-ACM → DBLP-Scholar. This is due to the skewed and bi-modal data distributions found with ER.

Both the *Coral* and *TCA* baselines are prone to differences in data distributions because they apply feature transformations to both the source and target data sets. As the ablation study in Section 5.4 will show, we can see that the MB and MSD data sets have a large difference in their conditional probability distributions. Because the marginal probability distributions are very similar in these data sets, *Coral* outperforms *TransER* and achieves better results in this scenario. However, in all other scenarios where we have a significant difference in marginal probability distributions, *TransER* outperforms *Coral* by on average 15% of F^* and 10% of *F1*.

5.2.2 Runtime Comparison. Table 3 shows runtimes of *TransER* and the compared baselines. Due to the high memory consumption of *TCA* we were not able to conduct experiments on the larger data sets with this method. Similarly, not all experiments with *DTAL** and *LocIT** completed within the 72 hours runtime limitation. The *Naive* baseline has the best runtimes as it simply applies the classifier trained on the source data set to the target data set. Next is *Coral* which requires less computations compared to the other baselines as it aligns second order statistics on the data sets.

Table 3: Feature matrix sizes (number of record pairs) and runtime results (in seconds) for *TransER* and baselines. 'ME' and 'TE' again refer to experiments exceeding the limitations of using over 200 GBytes of memory or runtimes of more than 72 hours.

Source ($\mathcal{D}^S$)	$\lvert\mathbf{X}^S\rvert$	Target ($\mathcal{D}^T$)	$\lvert\mathbf{X}^T\rvert$	*TransER*	Naive	DTAL* [28]	DR [54]	LocIT* [56]	TCA [44]	Coral [52]
DBLP-ACM	6,660	DBLP-Scholar	16,041	12	**1**	11,381	286	12,676	4,717	**1**
DBLP-Scholar	16,041	DBLP-ACM	6,660	20	**2**	11,651	493	1,968	4,566	**2**
MSD	27,544	MB	91,143	54	**4**	115,111	987	TE	ME	6
MB	91,143	MSD	27,544	221	**58**	128,108	867	TE	ME	104
IOS Bp-Dp	115,986	KIL Bp-Dp	242,457	621	**151**	TE	8,765	TE	ME	167
KIL Bp-Dp	242,457	IOS Bp-Dp	115,986	2,716	**1,139**	TE	36,001	TE	ME	2,223
IOS Bp-Bp	249,396	KIL Bp-Bp	406,038	1,650	**1,061**	TE	33,608	TE	ME	1,255
KIL Bp-Bp	406,038	IOS Bp-Bp	249,396	**2,623**	3,076	TE	25,160	TE	ME	5,403

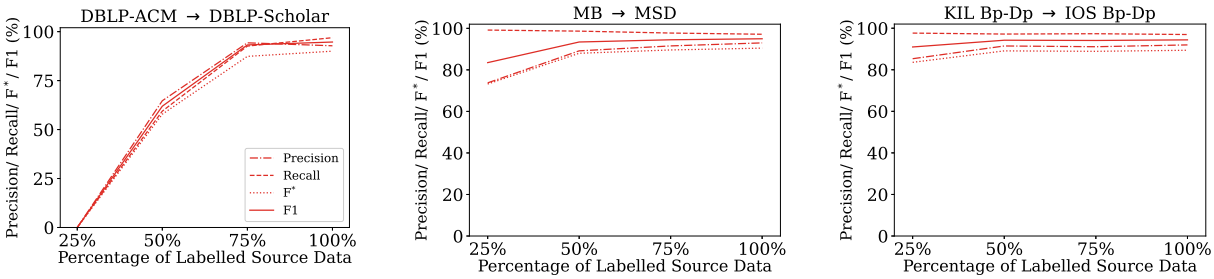

Figure 6: Precision, Recall, F^*, and *F1* sensitivity of *TransER* with regard to source data labelling, as discussed in Section 5.2.

The third best runtimes can be seen for *TransER* as it is computationally more efficient than the other baselines. However, we can see that *TransER* outperforms both the *Naive* and *Coral* baselines for the largest source data set with the KIL Bp-Bp → IOS Bp-Bp scenario. These generally lower runtimes for larger data sets for *TransER* are due to it using a reduced number of feature vectors because of the instance selection phase. Fourth is the *DR* baseline that employs an existing FastText [5] model to obtain the distributed representations of instances. Both *LocIT** and *TCA* are less efficient than *TransER*. *LocIT** consumes much time in generating the training samples for the supervised classifier it is using to select instances, while *TCA* consumes much time for the matrix computations used in feature transformations. The largest runtimes can be seen for *DTAL** as it is training deep transfer models for the ER classification step. This is a known issue in the literature as discussed by Mudgal et al. [38]. Overall, the runtimes of *TransER* are comparatively much lower than for the other baselines.

5.2.3 Sensitivity to Labelled Source Data Set Size. Due to limited space, we present the results of the remaining experiments on three data set pairs only, one bibliographic (DBLP-ACM → DBLP-Scholar), one music (MB → MSD), and one demographic (KIL Bp-Dp → IOS Bp-Dp) pair. Note that these data sets represent different data set sizes giving us insight into the behaviour of *TransER* when applied on data sets of different sizes.

Due to the increasingly large sizes of data sets to be employed in ER, finding a completely labelled data set for the source domain can be costly. Therefore, in Figure 6 we show how *TransER* behaves when different percentages of labelled data are available in the source domain. We increase the amount of labelled source data starting at 25% and increasing the size by 25% until we utilise the completely labelled source data sets. As the figure shows, the performance of *TransER* improves as the labelled source data set size increases. As can be seen, DBLP-ACM → DBLP-Scholar has very low linkage quality results for smaller sizes of labelled data whereas the other two data sets already have high linkage quality. This is to be expected because the DBLP-ACM data set is fairly small. Since the MB and KIL Bp-Dp data sets are larger, *TransER* provides good performance even with less labelled training data.

5.3 Parameter Sensitivity Analysis

Figure 7 shows how *TransER* is robust to the parameters t_c, t_l, t_p, and k, by varying each parameter while keeping the other parameters at their defaults, where we discuss default values below.

We first vary the threshold for instance confidence, t_c, in the range of [0.5, 1.0], where 1.0 indicates that only instances with a confidence similarity of $t_c = 1.0$ are being transferred. We can see that both F^* and *F1* results are fairly robust in this range with very little variations due to data set characteristics such as conditional probability distributions in the selected instances. Therefore, we choose $t_c = 0.9$ as the default that works well for all data sets.

Second, we vary the threshold for instance structural similarity, t_l, in the range of [0.5, 1.0] where $t_l = 1.0$ indicates only instances with a zero distance between their source and target neighbourhoods are being transferred. As we can see, with $0.5 \leq t_l \leq 0.9$ the overall F^* and *F1* results improve because higher thresholds drop instances with class conditional distribution differences. This improvement is clearly visible for DBLP-ACM → DBLP-Scholar as these are small data sets. However, further increasing t_l leads to dramatic drops is results because stricter thresholds remove many instances despite of their class conditional differences. Therefore, we set $t_l = 0.9$ as a default choice that works well for all data sets.

We then vary the threshold for pseudo label confidence, t_p, in the range of [0.5, 1.0]. From the figures we can see that *TransER* is

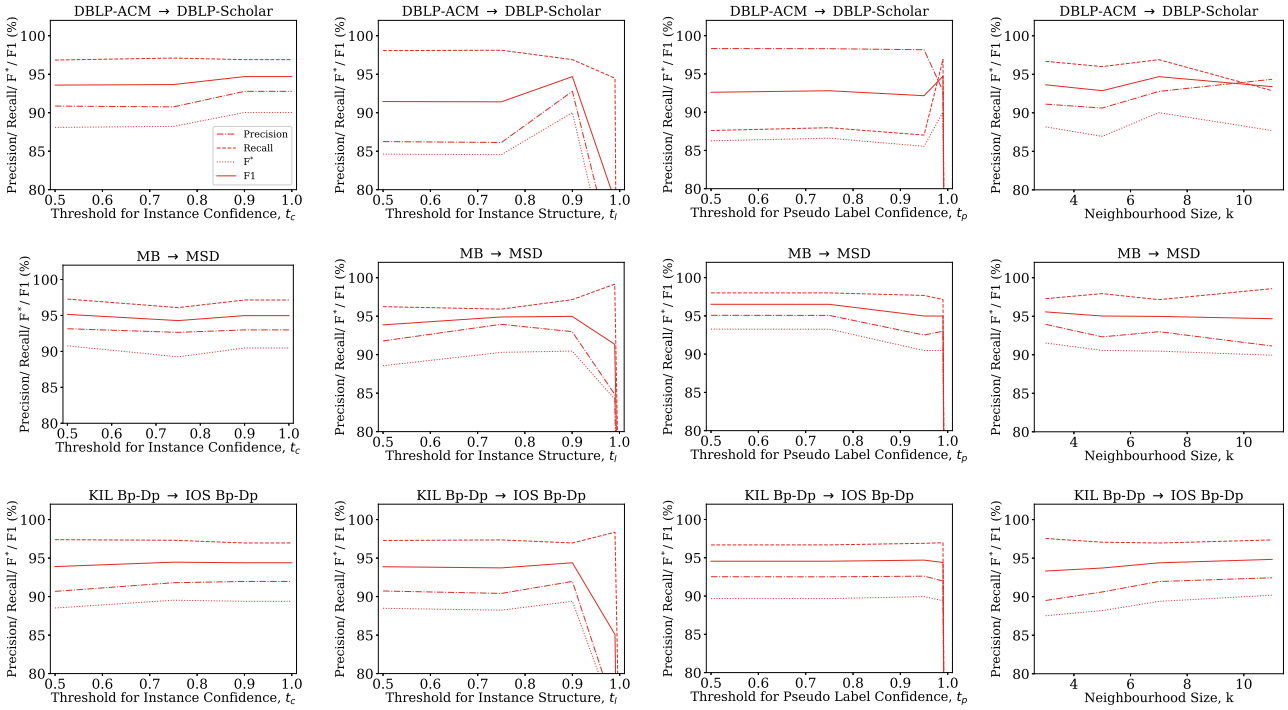

Figure 7: Precision, Recall, F^*, and *F1* sensitivity results of *TransER* for different t_c, t_l, t_p, and k values, as discussed in Section 5.3.

robust with $0.5 \leq t_p \leq 0.99$, but linkage quality drops with $t_p = 1.0$ because only a few instances are being transferred due to this strict filtering. For DBLP-ACM → DBLP-Scholar, the precision and recall results vary dramatically for $t_p > 0.95$ because of the low number of instances being transferred from the small DBLP-ACM data sets. Therefore, we set $t_p = 0.99$ as the default value that works well for all data sets including small data sets such as DBLP-ACM.

As shown in Figure 7, *TransER* is fairly robust to the neighbourhood size, k, in the range of $[3, 11]$. However, small data sets such as DBLP-ACM → DBLP-Scholar are more sensitive to different values of k because larger neighbourhoods for these smaller data sets are more ambiguous. This leads to larger variations in the confidence similarity scores calculated using Equation (1). However, when data set sizes increase from MB → MSD to KIL Bp-Dp → IOS Bp-Dp, then these variations reduce and the results become more robust. We therefore set the smallest neighbourhood size that performs well as the default choice, $k = 7$.

5.4 Ablation Analysis

Table 4 shows the contributions of each key component of *TransER*. In Section 4 we described the three phases of our framework, *SEL*, *GEN*, and *TCL*. As *TCL* uses the pseudo labels generated by *GEN*, we consider those two phases as a single component in this analysis showing results without *SEL* and without *GEN* and *TCL*, separately. Furthermore, in *SEL*, we use two different filtering thresholds to select instances. We therefore also show results by removing each filtering criterion, sim_c and sim_l, from *SEL*. Finally, *TransER* + sim_v corresponds to another filtering added to the instance selector phase, sim_v, the covariance similarity employed in *LocIT* [56].

When we remove *GEN* and *TCL* from *TransER*, we directly train a classifier on the selected instances instead of training a second classifier on the target. We can see that for all data sets either precision or recall is reduced along with a drop of F^* of up to 6% because we do not address the differences in the marginal probability distributions. However, we can see a smaller drop in the results for the MB → MSD scenario because both of these data sets have quite similar marginal probability distributions.

The *SEL* phase adjusts the differences in conditional probability distributions (the conflicting labels), as we discussed in Section 4.1. Without *SEL* we can see a drastic drop of linkage quality (65% in F^*) for the MB → MSD scenario. Table 1 shows the highest percentage of ambiguous feature vectors (22%) for MB, while MB and MSD have 64% of the ambiguous feature vectors in common. These account for different labels resulting in a large difference in conditional probability distributions and a drastic drop of results for the MB → MSD scenario. Similarly, the other scenarios also have results that are lower by up-to 41% for F^* and up-to 31% for *F1*.

The *SEL* phase filters instances based on the confidence, sim_c, and structural, sim_l, similarities from Equations (1) and (2). When we remove filtering by sim_c, Table 4 shows a drop of F^* for all data sets by up to 40%. This validates that source instances with lower confidence of their class labels account for the class conditional distribution difference. When we remove filtering by sim_l, F^* drops by up to 5% because we transfer source instances that have a different local distribution in the target domain.

Finally, when covariance similarity, sim_v, based filtering [56] is also applied in the *SEL* phase, the results are almost the same for both small and mid-sized data sets. This is because the covariance is not able to identify any further instances that have conflicting labels in two domains. As we consider small neighbourhood

Table 4: Ablation analysis for *TransER* that shows how each key component in the framework affects linkage quality, as discussed in Section 5.4.

Source	Target		TransER	without *GEN & TCL*	without *SEL*	without sim_c	without sim_l	TransER + sim_v
DBLP-ACM	DBLP-Scholar	*P*	92.78 ± 5.13	98.83 ± 0.52	85.06 ± 10.00	86.29 ± 9.18	86.20 ± 9.17	92.78 ± 5.13
		R	96.90 ± 1.27	86.17 ± 2.95	98.42 ± 1.35	98.18 ± 1.59	98.16 ± 1.52	96.90 ± 1.27
		F^*	**90.02 ± 4.31**	85.29 ± 2.63	83.74 ± 9.07	84.74 ± 8.15	84.66 ± 8.17	90.02 ± 4.31
		F1	**94.69 ± 2.46**	92.04 ± 1.56	90.88 ± 5.39	91.53 ± 4.83	91.48 ± 4.84	94.69 ± 2.46
MB	MSD	*P*	92.99 ± 2.19	91.95 ± 2.47	26.73 ± 38.68	82.45 ± 13.53	92.03 ± 3.63	92.99 ± 2.19
		R	97.15 ± 2.79	98.19 ± 0.51	31.54 ± 44.80	98.96 ± 0.62	94.56 ± 3.26	97.15 ± 2.79
		F^*	**90.47 ± 2.46**	90.40 ± 2.13	25.60 ± 37.23	81.64 ± 13.10	87.52 ± 5.30	90.47 ± 2.46
		F1	**94.98 ± 1.38**	94.95 ± 1.20	28.68 ± 40.95	89.29 ± 8.40	93.26 ± 3.15	94.98 ± 1.38
KIL-Bp-Dp	IOS-Bp-Dp	*P*	91.97 ± 1.62	85.20 ± 9.76	52.51 ± 15.78	54.24 ± 14.98	90.61 ± 2.28	91.85 ± 1.46
		R	96.96 ± 0.41	97.16 ± 0.70	80.70 ± 22.77	83.79 ± 19.01	97.33 ± 0.23	97.08 ± 0.26
		F^*	**89.39 ± 1.50**	83.00 ± 8.93	48.60 ± 17.71	49.24 ± 15.68	88.41 ± 2.02	89.38 ± 1.29
		F1	**94.39 ± 0.83**	90.43 ± 5.66	63.28 ± 17.89	64.47 ± 14.62	93.83 ± 1.13	94.39 ± 0.71

sizes (with a default of $k = 7$), the impact of covariance does not contribute to an improvement of results. Furthermore, we can see a slight drop of results for the large KIL Bp-Dp → IOS Bp-Dp data set pair, because some instances that do not have a class conditional distribution difference are dropped with this additional criterion.

5.5 Limitations

One limitation of *TransER* is its focus on structured data because it uses attribute based similarities to generate features. However, for unstructured data [26, 37, 58] the best option is to use deep learning TL methods for ER since they are better at representing unstructured (textual) data. Another limitation of our work, as with all existing TL methods for ER, is that it covers the homogeneous TL scenario only. If the source and target domains have two completely different feature spaces, then *TransER* cannot be applied to classify record pairs in the target domain. How to employ TL for ER in heterogeneous scenarios is an open research question [26].

6 CONCLUSION AND FUTURE WORK

We have presented a novel framework for TL for ER that includes three key phases: instance selection, pseudo label generation, and target domain classification. The first phase addresses the class conditional distribution difference in two domains while the latter two phases address the marginal probability distribution difference. We minimise the effect of class imbalance and bimodal distributions for ER data sets by selecting instances for transferring that have similar local neighbourhoods. We have shown that our framework can outperform several state-of-the-art TL methods, and that it is one the most efficient TL methods among the baselines compared. Our framework is also robust to parameter settings and each key component substantially contributes to high ER quality.

We plan to extend our framework in four ways: explore how to support heterogeneous TL for ER where the data sets have non-common feature spaces, investigate how to perform TL when some labels are available in the target domain, explore how to choose the best source domain when multiple semantically related labelled data sets are available, and explore how to integrate our framework with active learning techniques to improve its performance.

REFERENCES

[1] Nils Barlaug and Jon Atle Gulla. 2021. Neural Networks for Entity Matching: A Survey. *Transactions on Knowledge Discovery from Data* 15, 3 (2021), 37.

[2] Jon Louis Bentley. 1975. Multidimensional binary search trees used for associative searching. *Commun. ACM* 18, 9 (1975), 509–517.

[3] Indrajit Bhattacharya and Lise Getoor. 2007. Collective Entity Resolution in Relational Data. *Transactions on Knowledge Discovery from Data* 1, 1 (2007), 36.

[4] Mikhail Bilenko and Raymond J. Mooney. 2003. Adaptive Duplicate Detection Using Learnable String Similarity Measures. In *International Conference on Knowledge Discovery and Data Mining*. ACM, Washington, USA, 39–48.

[5] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics* 5 (2017), 135–146.

[6] Ursin Brunner and Kurt Stockinger. 2020. Entity matching with transformer architectures-a step forward in data integration. In *International Conference on Extending Database Technology*. Copenhagen, Denmark.

[7] Qun Chen, Zhaoqiang Chen, Youcef Nafa, Tianyi Duan, and Zhanhuai Li. 2020. Adaptive Deep Learning for Entity Resolution by Risk Analysis. *arXiv preprint arXiv:2012.03513* (2020).

[8] Qingchao Chen, Yang Liu, Zhaowen Wang, Ian Wassell, and Kevin Chetty. 2018. Re-weighted Adversarial Adaptation Network for Unsupervised Domain Adaptation. In *Conference on Computer Vision and Pattern Recognition*. IEEE, Utah, 7976–7985.

[9] Peter Christen. 2012. *Data Matching – Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Springer, Heidelberg.

[10] Peter Christen. 2016. Application of Advanced Record Linkage Techniques for Complex Population Reconstruction. *arXiv preprint arXiv:1612.04286* (2016), 12.

[11] Peter Christen, Thilina Ranbaduge, and Rainer Schnell. 2020. *Linking Sensitive Data: Methods and Techniques for Practical Privacy-Preserving Information Sharing*. Springer, Heidelberg.

[12] Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. 2007. Boosting for Transfer Learning. In *International Conference on Machine Learning*. ACM, New York, USA, 193–200.

[13] Sanjib Das, AnHai Doan, Paul Suganthan G. C., Chaitanya Gokhale, Pradap Konda, Yash Govind, and Derek Paulsen. 2021. The Magellan Data Repository. Retrieved August 19, 2021 from https://sites.google.com/site/anhaidgroup/useful-stuff/data

[14] Hal Daumé III. 2007. Frustratingly Easy Domain Adaptation. *Association of Computational Linguistics* 45 (2007), 256–263.

[15] Xin Dong, Alon Halevy, and Jayant Madhavan. 2005. Reference reconciliation in complex information spaces. In *International Conference on Management of Data*. ACM, Baltimore, 85–96.

[16] Xin Dong and Theodoros Rekatsinas. 2018. Data Integration and Machine Learning: A Natural Synergy. *VLDB Endowment* 11, 12 (2018), 2094–2097.

[17] Xin Dong and Divesh Srivastava. 2015. *Big Data Integration*. Morgan and Claypool Publishers.

[18] Uwe Draisbach, Peter Christen, and Felix Naumann. 2019. Transforming Pairwise Duplicates to Entity Clusters for High-quality Duplicate Detection. *Journal of Data and Information Quality* 12, 1 (2019), 1–30.

[19] Muhammad Ebraheem, Saravanan Thirumuruganathan, Shafiq Joty, Mourad Ouzzani, and Nan Tang. 2018. Distributed Representations of Tuples for Entity Resolution. *VLDB Endowment* 11, 11 (2018), 1454–1467.

[20] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. 2012. Geodesic Flow Kernel for Unsupervised Domain Adaptation. In *Conference on Computer Vision and Pattern Recognition*. IEEE, Rhode Island, USA, 2066–2073.

[21] Yash Govind, Erik Paulson, Palaniappan Nagarajan, Paul Suganthan G. C., AnHai Doan, Youngchoon Park, Glenn M. Fung, Devin Conathan, Marshall Carter, and Mingju Sun. 2018. Cloudmatcher: A Hands-off Cloud/Crowd Service for Entity Matching. *VLDB Endowment* 11, 12 (2018), 2042–2045.

[22] David J Hand and Peter Christen. 2018. A Note on Using the F-measure for Evaluating Record Linkage Algorithms. *Statistics and Computing* 28, 3 (2018), 539–547.

[23] David J Hand, Peter Christen, and Nishadi Kirielle. 2021. F*: An Interpretable Transformation of the F-measure. *Machine Learning* 110, 3 (2021), 451–456.

[24] Oktie Hassanzadeh, Fei Chiang, Hyun Chul Lee, and Renée J. Miller. 2009. Framework for Evaluating Clustering Algorithms in Duplicate Detection. *VLDB Endowment* 2, 1 (2009), 1282–1293.

[25] Jing Jiang and ChengXiang Zhai. 2007. Instance Weighting for Domain Adaptation in NLP. In *Association of Computational Linguistics*. Association of Computational Linguistics, Prague, Czech Republic, 264–271.

[26] Di Jin, Bunyamin Sisman, Hao Wei, Xin Luna Dong, and Danai Koutra. 2022. Deep Transfer Learning for Multi-source Entity Linkage via Domain Adaptation. *VLDB Endowment* (2022).

[27] Dmitri V. Kalashnikov and Sharad Mehrotra. 2006. Domain-independent data cleaning via analysis of entity-relationship graph. *Transactions on Database Systems* 31, 2 (2006), 716–767.

[28] Jungo Kasai, Kun Qian, Sairam Gurajada, Yunyao Li, and Lucian Popa. 2019. Low-resource Deep Entity Resolution with Transfer and Active Learning. In *Annual Meeting of the Association for Computational Linguistics*. ACL, Florence, 5851–5861.

[29] Pradap Konda, Sanjib Das, Paul Suganthan GC, AnHai Doan, Adel Ardalan, Jeffrey R Ballard, et al. 2016. Magellan: Toward Building Entity Matching Management Systems. *VLDB Endowment* 9, 12 (2016), 1197–1208.

[30] Pradap Konda, Sanjay Subramanian Seshadri, Elan Segarra, Brent Hueth, and AnHai Doan. 2019. Executing Entity Matching End to End: A Case Study. In *International Conference on Extending Database Technology*. Lisbon, 489–500.

[31] Hanna Köpcke and Erhard Rahm. 2010. Frameworks for Entity Matching: A comparison. *Data & Knowledge Engineering* 69, 2 (2010), 197–210.

[32] Hanna Köpcke, Andreas Thor, and Erhard Rahm. 2010. Evaluation of Entity Resolution Approaches on Real-World Match Problems. *VLDB Endowment* 3, 1–2 (2010), 484–493.

[33] Hye-Chung Kum, Ashok Krishnamurthy, Ashwin Machanavajjhala, and Stanley Ahalt. 2014. Social Genome: Putting Big Data to Work for Population Informatics. *Computer* 47, 1 (2014), 56–63.

[34] Database Group Leipzig. 2021. Benchmark Datasets for Entity Resolution. Retrieved August 19, 2021 from https://dbs.uni-leipzig.de/research/projects/object_matching/benchmark_datasets_for_entity_resolution

[35] Xuejun Liao, Ya Xue, and Lawrence Carin. 2005. Logistic Regression with an Auxiliary Data Source. In *International Conference on Machine Learning*. ACM, New York, USA, 505–512.

[36] Mingsheng Long, Jianmin Wang, Guiguang Ding, Jiaguang Sun, and Philip S. Yu. 2013. Transfer Feature Learning with Joint Distribution Adaptation. In *International Conference on Computer Vision*. IEEE, Sydney, Australia, 2200–2207.

[37] Michael Loster, Ioannis Koumarelas, and Felix Naumann. 2021. Knowledge Transfer for Entity Resolution with Siamese Neural Networks. *Journal of Data and Information Quality* 13, 1 (2021), 1–25.

[38] 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 *International Conference on Management of Data*. ACM, Houston, USA, 19–34.

[39] Charini Nanayakkara, Peter Christen, and Thilina Ranbaduge. 2019. Robust Temporal Graph Clustering for Group Record Linkage. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, Macau, 526–538.

[40] Felix Naumann and Melanie Herschel. 2010. *An Introduction to Duplicate Detection*. Morgan and Claypool Publishers.

[41] Sahand N. Negahban, Benjamin I.P. Rubinstein, and Jim Gemmell Gemmell. 2012. Scaling Multiple-Source Entity Resolution Using Statistically Efficient Transfer Learning. In *International Conference on Information and Knowledge Management*. ACM, Hawaii, USA, 2224–2228.

[42] Hao Nie, Xianpei Han, Ben He, Le Sun, Bo Chen, Wei Zhang, Suhui Wu, and Hao Kong. 2019. Deep Sequence-to-Sequence Entity Matching for Heterogeneous Entity Resolution. In *International Conference on Information and Knowledge Management*. ACM, Beijing, China, 629–638.

[43] Matteo Paganelli, Francesco Del Buono, Pevarello Marco, Francesco Guerra, and Maurizio Vincini. 2021. Automated Machine Learning for Entity Matching Tasks. In *International Conference on Extending Database Technology*. Nicosia, 325–330.

[44] Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. 2011. Domain Adaptation via Transfer Component Analysis. *Transactions on Neural Networks* 22, 2 (2011), 199–210.

[45] Sinno Jialin Pan and Qiang Yang. 2009. A Survey on Transfer Learning. *Transactions on Knowledge and Data Engineering* 22, 10 (2009), 1345–1359.

[46] George Papadakis, Ekaterini Ioannou, Emanouil Thanos, and Themis Palpanas. 2021. *The Four Generations of Entity Resolution*. Morgan and Claypool Publishers.

[47] George Papadakis, Dimitrios Skoutas, Emmanouil Thanos, and Themis Palpanas. 2020. Blocking and Filtering Techniques for Entity Resolution: A Survey. *Computing Surveys* 53, 2 (2020), 1–42.

[48] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, et al. 2011. Scikit-Learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011).

[49] Kun Qian, Lucian Popa, and Prithviraj Sen. 2017. Active Learning for Large-Scale Entity Resolution. In *Conference on Information and Knowledge Management*. ACM, Singapore, 1379–1388.

[50] Alice Reid, Ros Davies, and Eilidh Garrett. 2002. Nineteenth-Century Scottish Demography from Linked Censuses and Civil Registers: A 'Sets of Related Individuals' Approach. *History and Computing* 14, 1–2 (2002), 61–86.

[51] Michael Stonebraker, Daniel Bruckner, Ihab F Ilyas, George Beskales, Mitch Cherniack, Stanley B Zdonik, Alexander Pagan, and Shan Xu. 2013. Data Curation at Scale: The Data Tamer System. In *Biennial Conference on Innovative Data Systems Research*. Asilomar, California, 10.

[52] Baochen Sun, Jiashi Feng, and Kate Saenko. 2016. Return of Frustratingly Easy Domain Adaptation. In *Conference on Artificial Intelligence*. AAAI, Arizona, 2058–2065.

[53] Qian Sun, Rita Chattopadhyay, Sethuraman Panchanathan, and Jieping Ye. 2011. A Two-Stage Weighting Framework for Multi-Source Domain Adaptation. In *International Conference on Neural Information Processing Systems*. Curran Associates Inc., Granada, Spain, 505–513.

[54] 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).

[55] Jesper E Van Engelen and Holger H Hoos. 2020. A Survey on Semi-Supervised Learning. *Machine Learning* 109, 2 (2020), 373–440.

[56] Vincent Vercruyssen, Wannes Meert, and Jesse Davis. 2020. Transfer Learning for Anomaly Detection through Localized and Unsupervised Instance Selection. In *Conference on Artificial Intelligence*. AAAI, New York, USA, 6054–6061.

[57] Jiannan Wang, Tim Kraska, Michael J. Franklin, and Jianhua Feng. 2012. CrowdER: Crowdsourcing Entity Resolution. *VLDB Endowment* 5, 11 (2012), 1483–1494.

[58] Chen Zhao and Yeye He. 2019. Auto-EM: End-to-end Fuzzy Entity-Matching using Pre-trained Deep Models and Transfer Learning. In *The World Wide Web Conference*. ACM, San Francisco, USA, 2413–2424.