

# Efficient Model Repository for Entity Resolution: Construction, Search, and Integration

Victor Christen  
 Leipzig University & ScaDS.AI  
 Leipzig, Germany  
 christen@informatik.uni-leipzig.de

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

## Abstract

Entity resolution (ER) is a fundamental task in data integration that enables insights from heterogeneous data sources. The primary challenge of ER lies in classifying record pairs as matches or non-matches, which in multi-source ER (MS-ER) scenarios can become complicated due to data source heterogeneity and scalability issues. Existing methods for MS-ER generally require labeled record pairs, and such methods fail to effectively reuse models across multiple ER tasks. We propose *MoRER (Model Repositories for Entity Resolution)*, a novel method for building a model repository consisting of classification models that solve ER problems. By leveraging feature distribution analysis, *MoRER* clusters similar ER tasks, thereby enabling the effective initialization of a model repository with a moderate labeling effort. Experimental results on three multi-source datasets demonstrate that *MoRER* achieves comparable or better results to methods that have label-limited budgets, such as active learning and transfer learning approaches, while outperforming self-supervised approaches that utilize large pre-trained language models. When compared to supervised transformer-based methods, *MoRER* achieves comparable or better results, depending on the size of the training data set used.

## Keywords

Record Linkage, Model Reuse, Distribution Analysis

## 1 Introduction

Due to the increasing number of available data sources, data integration is essential for comprehensive analysis and a standardized view of information that is often scattered across sources [41]. A cornerstone of data integration is entity resolution (ER), which identifies and links records that refer to the same real-world entity [3]. The broad range of potential data quality issues and data source heterogeneity make ER challenging [3, 6]. Multi-source ER [6], especially, amplifies multiple challenges that range from scalability issues to achieving high-quality linkage results.

The quality of the ER result between two data sources is highly dependent on the classification step [41]. This step classifies record pairs into matches and non-matches based on attribute value similarities or derived embeddings from pre-trained language models. State-of-the-art methods use machine learning (ML) based approaches since they have been shown to achieve high-quality results [9]. However, the majority of ML methods require labeled training data to build a classification model, where generating such data can be a time-consuming and expensive task [28].

Various methods have been proposed to reduce the labeling effort, such as active learning (AL) [5, 27, 29, 33] and transfer

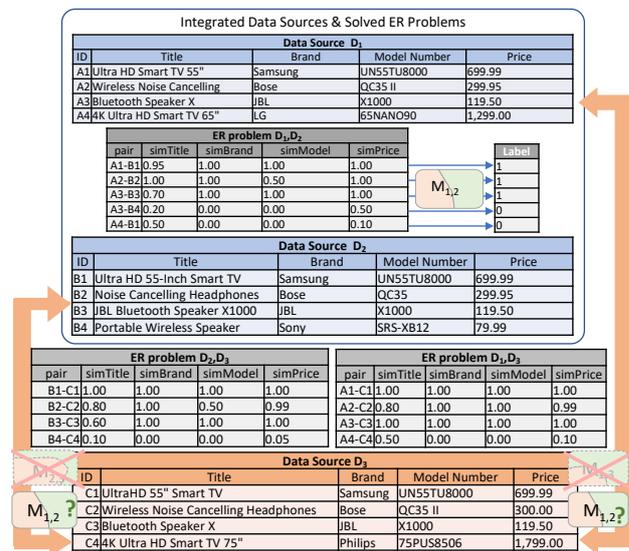


Figure 1: Motivation of reusing solved ER tasks for new tasks. The data sources  $D_1$  and  $D_2$  are already linked utilizing similarity feature vectors and a model  $M_{1,2}$  to label each record pair. The question is whether the derived model  $M_{1,2}$  can also be applied to the new data source,  $D_3$ , to match  $D_3$  to  $D_1$  and  $D_2$ , or if new models have to be generated.

learning [17, 18, 25]. However, to the best of our knowledge, no existing method supports the reuse of trained ER classifiers by building and utilizing a model repository for ER.

Typical use cases for model repositories are found in research institutions and companies that require the continuous integration of new and diverse data sources. For example, in healthcare, hospitals and clinics often need to integrate patient records from various systems to create unified medical histories in order to enable better diagnosis, treatment, and research [30]. Similarly, in census data analysis, government agencies must reconcile records from multiple surveys and administrative sources to ensure accurate population statistics and policy planning [23]. In e-commerce, product comparison portals and marketplaces integrate product catalogs from different vendors to provide search and recommendation functionalities for users.

In all these scenarios, our method will allow organizations to reuse previously trained models from solved ER tasks, significantly reducing both labeling efforts and computational costs. Furthermore, our method lays the groundwork for establishing ER matching services, enabling users to solve any ER problem by leveraging existing models. This capability is particularly valuable in dynamic environments where new data sources are continuously added, ensuring scalability and efficiency in real-world applications.

EDBT '26, Tampere (Finland)

© 2026 Copyright held by the owner/author(s). Published on OpenProceedings.org under ISBN 978-3-98318-104-9, series ISSN 2367-2005. Distribution of this paper is permitted under the terms of the Creative Commons license CC-by-nc-nd 4.0.

Fig. 1 illustrates the motivation of our proposed method. Assuming we have two integrated data sources,  $D_1$  and  $D_2$ , consisting of products specified by the shown attributes. The integration of these data sources requires the solution of the ER problem given by the classification task of the similarity feature vectors into matches and non-matches. These similarity features are derived from the attribute-wise comparisons between record pairs. Therefore, the classification model,  $M_{1,2}$ , based on labeled record pairs, is trained and used to classify the record pairs  $A1 - B1, A2 - B2, A3 - B3, A3 - B4$  and  $A4 - B1$ . Later on, a new data source  $D_3$  needs to be integrated, where we calculate similarity features between data sources  $D_1$  and  $D_3$ , as well as between  $D_2$  and  $D_3$ .

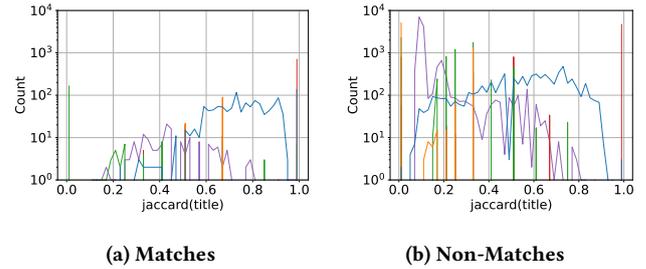
In a naive solution, we would build one new classification model,  $M_{1,3}$  and  $M_{2,3}$ , for each new ER task between the data sources  $D_1$  and  $D_3$  and  $D_2$  and  $D_3$ , respectively. Due to the increasing number of models with the growing number of integrated data sources, the labeling effort for generating training data would, however, increase substantially. Instead of building the new models  $M_{1,3}$  and  $M_{2,3}$ , our aim is to reuse the existing model  $M_{1,2}$  that has already solved the (potentially similar) ER task between  $D_1$  and  $D_2$ . As the amount of data to be integrated increases, it becomes crucial to efficiently generate training data for similar ER problems to ensure effective and scalable classification. Our proposed method, *MoRER*, addresses these challenges.

An alternative approach would be to unify all existing ER problems to create a comprehensive training dataset. However, individual ER problems can have different distributions of their similarity features. This would make it challenging for a single classification model to account for the potentially diverse similarity distributions. To overcome this issue, the idea would be to create new training data for each new ER problem and combine these with existing one. However, such an approach is not scalable, since it requires the training data generation for each new ER problem. Fig. 2 illustrates the variations in similarity distributions calculated by the Jaccard string comparison function [3] on the title for products on the WDC computer dataset [34], which consists of four data sources. Given these very diverse similarity distributions, a single unified classification model would struggle to capture the diverse characteristics of matches and non-matches. Moreover, new ER problems might not fit into an existing model, so retraining is required.

To the best of our knowledge, no current approach takes model reuse into account. Existing approaches assume that training data are available to train a model and that this model can be used for all future record pairs, as in progressive or incremental ER [12].

Therefore, we suggest a method to create an ER model repository and to reuse the maintained models for similar future ER problems. The reuse of models is highly relevant where new data sources must be integrated in target data sources such as data warehouses or for ad hoc analysis with data from data lakehouses. In this paper, we make the following contributions:

- We propose a novel method to build an ER model repository of classification models for ER problems. Our method analyzes the similarity feature space to determine the most appropriate model for an unsolved novel ER problem. Unlike existing ER methods, our method distinguishes between ER tasks with different similarity feature spaces and selects the appropriate models for the various ER tasks.
- Our proposed method efficiently initializes the repository with models that require only little labeling effort. To do



**Figure 2: Example of the similarity distributions calculated using the Jaccard string comparison function [3] for title considering the ER problems in the WDC computer data set. Each of the five lines represents an ER problem.**

so, we build an ER problem graph and cluster ER tasks according to their similar feature spaces. For each cluster, we build a classification model using the most relevant similarity feature vectors as training data. The resulting clusters and models can be stored in a backend system and used for future ER tasks.

- We evaluate our method using three multi-source ER datasets with up to 23 sources and compare it with state-of-the-art AL methods for multi-source ER, a transfer learning method, and pre-trained language model-based ER methods, considering the effectiveness and efficiency of all these methods.

The remainder of this paper is organized as follows. In Section 2 we formalize the problem we aim to solve, and in Section 3 we review related work on label-efficient and state-of-the-art ER methods. In Section 4 we then describe our method in detail, which we evaluate in Section 5 and compare it with several state-of-the-art approaches. In Section 6, we discuss our results and provide recommendations. We conclude in Section 7 with future research directions.

## 2 Problem Definition

Our method aims to solve ER problems by using classification models from a repository. In real-world scenarios, continuous integration of new data sources is commonly required for ongoing analysis and maintenance [14].

Following the notation shown in Table 1, we are given a set  $\mathcal{D}$  of data sources  $\{D_1, \dots, D_z\}$ . An ER problem  $p_{k,l}$  is defined for a pair of data sources  $D_k$  and  $D_l$  consisting of similarity features for each record pair. We utilize an initial set  $\mathcal{P}_I$  of ER problems  $p_{i,l}$  to set up the ER model repository. We aim to solve new ER problems  $p_{i,l} \in \mathcal{P}_U$  utilizing the initial ER problems  $\mathcal{P}_I$ .

An ER problem  $p_{i,l}$  is defined for a data source pair  $D_i, D_l \in \mathcal{D}$  and consists of a set of similarity feature vectors  $w$ . Each similarity feature vector  $w \in p_{k,l}$  represents a record pair, where each element of  $w$  corresponds to a similarity in the range of  $[0,1]$ , calculated based on attribute value comparisons using a similarity function. We denote an element of a similarity feature vector  $w$  as feature  $f$ . We assume that even for heterogeneous data sources, common features  $f$  exist based on common attributes between the different data sources, which are significant to describe the entities of a certain domain, such as the title of products, price information, etc. In the example in Fig. 1,  $\mathcal{P}_I$  consists of the ER problem  $P_{D_1,D_2}$  and  $\mathcal{P}_U$  includes the ER problems  $P_{D_2,D_3}$  and  $P_{D_1,D_3}$ .

Symbol	Description
$\mathcal{D}$	Set of data sources $D_1, D_2, \dots, D_z$
$\mathcal{P}_I$	Initial ER problems
$\mathcal{P}_U$	Unsolved ER problems
$p_{i,k}$	Set of similarity feature vectors $w$ between $D_i$ and $D_k$
$G_P$	ER problem similarity graph
$C_P$	Set of clusters consisting of similar ER problems $p_{i,k}$
$f$	Feature representing a similarity function applied on corresponding attributes
$w$	Similarity feature vector consisting of similarities
$d_{k,l}^f$	Distribution of the feature $f$ from similarity feature vectors $w \in p_{k,l}$
$CDF_{k,l}^f$	Cumulative distribution function for feature $f$ between data sources $D_k$ and $D_l$

**Table 1: Description of used symbols.**

To reduce the effort of building new classification models and the underlying label generation process, our solution groups similar ER problems  $p_{k,l} \in \mathcal{P}_I$ . We determine the similarity between pairs of ER problems utilizing univariate distribution tests to group similar problems into clusters  $C_P$ . We hypothesize that the ER tasks within a certain cluster  $C \in C_P$  can be effectively classified using a unified classification model  $M_C$ .

The analysis uses the similarity distribution  $d_{k,l}^f$  and the cumulative distribution function  $CDF_{k,l}^f$  for each feature  $f$  based on a data source pair  $D_k$  and  $D_l$ . We construct an ER problem similarity graph  $G_P = (\mathcal{P}_I \cup \mathcal{P}_U, E)$  where the edges represent the determined similarities between ER problems. The graph is used for clustering and is extendable to integrate new ER problems of  $\mathcal{P}_U$ .

**DEFINITION 1 (ENTITY RESOLUTION MODEL SEARCH).** *We define the problem of solving ER problems  $p_{D_x, D_z} \in \mathcal{P}_U$  as the search for a similar cluster  $C \in C_P$  to apply the model  $M_C$  on the ER problem between  $D_x$  and  $D_z$ . Moreover, we update the identified model  $M_C$  if the ER problems of cluster  $C$  become dissimilar or the coverage of training data of a certain cluster decreases below a certain threshold.*

### 3 Related Work

In this section, we discuss state-of-the-art ER approaches from traditional machine learning (ML) based methods to language model based solutions such as utilizing LLMs. In general, the main goal of our method is to reduce the effort for the model generation step. Therefore, we discuss budget-limited ML methods in ER, such as self-supervised, active, and transfer learning.

ER methods classify record pairs based on features derived from similarity functions [3]. In the past, supervised ER techniques have been developed, leveraging both traditional ML algorithms [1, 19, 20] and deep learning-based methods [10, 28]. The advent of transformer-based language models [44] has further advanced ER by enabling the representation of textual and unstructured data as embeddings, which can be directly compared [2, 22]. Transformers, powered by self-attention mechanisms, process input data in parallel, making them highly effective for tasks like natural language processing and sequence modeling. State-of-the-art methods, such as *Ditto* [22], fine-tune transformer models like BERT [8] and DistilBERT [38] to classify record pairs as

matches or non-matches. Auto-EM [52] relies on pre-trained models for different entity types to classify value pairs as match or non-match. For a new ER problem, the relevant entity types are identified, allowing the corresponding models to be utilized for classifying the record pairs. *Unicorn* [11] generates an universal model addressing various matching tasks such as entity alignment, ontology matching, or ER, by using a unified encoder model and adopting a mixture-of-experts model. *MultiEM* [50] clusters records from multiple data sources utilizing embeddings from pre-trained language models. This approach follows a hierarchical merge strategy where two data sources are compared to avoid the comparison of all data source pairs. While the proposed approach shows its strength in efficiency, it requires the identification of effective hyperparameters, the similarity threshold  $m$ , and clustering-related parameters. Peeters et al. [32] showed the feasibility of using large language models (LLMs) for ER on various real-world datasets and different prompt variations without any fine-tuning. Recent work [51] achieved comparable results by using problem-dependent fine-tuned small language models [51] with a significant cost reduction.

**Self-supervised Learning.** Sudowoodo [47] employs contrastive self-supervised learning to learn a similarity-aware data representation model for entities. The objective is to position similar entities closely together in the representation space while ensuring that dissimilar ones are placed further apart. The primary advantage of this approach is that it allows for the creation of meaningful data representations without the need for extensive labeled datasets. The fine-tuned model can be effectively utilized for various downstream tasks such as ER. ZeroER [49] aims to learn the probability density functions of matches and non-matches based on the similarity features. The approach uses an adapted Gaussian Mixture model to determine the distributions of these similarity feature vectors for both classes (matches and non-matches). The adaptations address the overfitting issue and the transitivity constraint occurring in ER.

**Active Learning.** AL approaches aim to minimize the labeling effort that is particularly important for multi-source ER due to the numerous classification tasks involved. Most AL methods focus on pairwise ER problems [5, 27, 29, 46] utilizing uncertainty measures and distances between feature vectors. Only one study [33] (*Almsr*) has specifically addressed multi-source ER by leveraging similarity graphs generated from the ER process. This work employs the concept of transitive closure to identify informative links, enabling the detection of potential false negatives and false positives. Specifically, it assesses whether two records, classified as non-matches, are connected through the transitive closure. Potential false positives are identified using the minimum cut of the graph. At the same time, the resulting graph also facilitates the identification of false negatives, represented as missing edges among record pairs within connected components.

**Transfer Learning.** Transfer learning methods aim to adapt a source domain to a target domain, minimizing the need for model training and data generation in the target domain. In the context of ER, these methods leverage existing models and training data from labeled ER tasks. For instance, Jin et al. [17] developed a joint neural network with a shared attention mechanism to optimize the weighting of different attribute comparisons for both source and target ER tasks. Loster et al. [25] proposed a method for transferring knowledge of a trained Siamese network to a target domain. The Siamese network learns record embeddings based on the given attributes utilizing a bidirectional LSTM

network (BLSTM) [39]. The architecture allows the transfer of embeddings and weights from the BLSTM layers encoding the attributes. Kirielle et al. [18] et al. proposed a method for labeling record pairs in unsolved ER tasks, thereby generating a classification model suitable for traditional ER tasks. The authors identify similar feature vectors between source and target entity resolution tasks and utilize the labels from similar target vectors to construct the target model.

**Limitations** Despite substantial progress in ER through machine learning, active learning, and transfer learning, existing approaches face several significant limitations, particularly in multi-source and dynamic integration settings. Most existing approaches focus on pairwise ER, requiring the generation of new classification models for each ER problem when the similarity feature space changes, resulting in a high training effort. In the following, we summarize the specific limitations for each area.

Most AL approaches are tailored to pairwise ER and do not generalize to the complexities of multi-source settings, where the number of possible classification tasks grows rapidly. Even recent multi-source AL methods, such as *Almser* [33], focus on optimizing label selection within known tasks but lack mechanisms to determine when existing models can be reused or when new models (and thus fresh training data) are necessary for new ER problems.

Transfer learning methods are limited by their inability to effectively select the most appropriate source task(s) when multiple candidates exist. Most transfer learning approaches require significant similarity between source and target domains, which is difficult to guarantee in heterogeneous and evolving multi-source environments. Furthermore, existing methods do not provide systematic procedures for evaluating model suitability.

While pretrained language model-based and LLM methods have substantially advanced ER performance, they struggle with practical constraints. Fine-tuning such models requires large labeled datasets, making them costly and impractical in scenarios with limited annotation resources. Moreover, LLM approaches are monetarily intensive and struggle to scale to the massive number of record pairs present in real-world multi-source ER tasks.

No current method fully supports the efficient reuse of both models and labeled data across multiple, evolving ER tasks. In scenarios where data sources and their characteristics frequently change, existing approaches either require retraining from scratch or risk applying poorly matched models, leading to degraded performance. Methods that attempt to unify training data from multiple ER problems ignore the heterogeneity in feature distributions, which limits model generalizability and scalability.

## 4 Reusing of Solved ER Tasks

In this section, we introduce our novel method *MoRER* that takes advantage of previously solved ER problems, denoted as  $\mathcal{P}_I$ , along with their corresponding classification models. We initially give an overview and then describe the steps of *MoRER* in detail.

### 4.1 Overview of *MoRER*

The input of our method is a set of ER problems that are used to initialise the repository. Our main focus is on reusing ER problems. As a reminder, we briefly describe how the similarity feature vectors  $w \in p_{k,l}$  can be determined. Theoretically, various blocking techniques [31] can be employed to reduce the search space based on the similarity calculations used. Recent approaches

utilize pre-trained language models to generate embeddings representing records [22, 50]. In this case, nearest-neighbour search approaches [26, 42] can be used to reduce the number of candidate record pairs [31].

The similarities can be computed using appropriate similarity functions tailored to the relevant attributes of the records being compared [3]. A further strategy to leverage data sources with different attributes is the use of pre-trained language models. The models can generate embeddings that encompass all attributes, functioning similarly to a bag-of-words approach. Using this strategy, an ER can calculate embedding similarities for certain or all attributes.

Figure 3 shows the workflow to efficiently create a model repository to solve new ER tasks of  $\mathcal{P}_I$ . We first conduct distribution analysis to determine whether the similarity distributions of features  $f_1, \dots, f_i$  from the ER problems are similar considering univariate and multivariate analysis tests. We calculate an similarity score,  $sim_p$ , for each  $p_{k,l} \in \mathcal{P}_I$ .

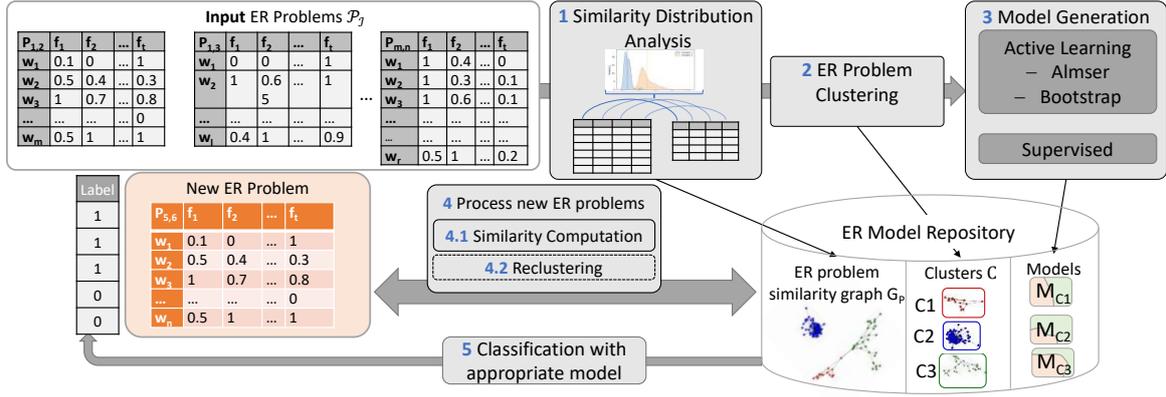
From the pairwise comparison of ER problems, we derive the ER problem similarity graph  $G_P$ . The ER problems represent the vertices, and the similarities between the ER problems are the weighted edges. We cluster the graph  $G_P$  using the Leiden algorithm [43]. In pre-experiments, we also evaluated other algorithms such as Girvan-Newmann [13] and label propagation [35], leading to similar results. As shown in Fig. 3, we calculate pairwise the similarities between  $P_{1,2}, P_{1,3} \dots P_{m,n}$  and compute clusters based on the graph  $G_P$  resulting in the clusters  $C_1, C_2$ , and  $C_3$ .

We consider all ER tasks  $p_{k,l} \in C^i$  as similar, assuming that one model is sufficient to classify the underlying similarity vectors from these ER tasks. Due to this simplification, we only need training data to generate one model for each cluster  $C^i \in \mathcal{C}_P$ .

The supervised solution in the model generation step considers all feature vectors of ER problems in  $\mathcal{P}_I$  as training data, requiring high labeling effort. To reduce labeling effort, we apply active learning for each group to determine labeled training data [27, 33]. The budget  $b_{tot}$  constrains the total number of labeled data. Due to the skew of the size of clusters, a uniform distribution of the budget regarding the clusters is not useful. Therefore, we proportionally distribute the budget based on the number of similar feature vectors of the clusters. Moreover, we handle singleton clusters separately since we want to prioritize clusters with more than one ER problem.

The result of the AL procedure for a cluster  $C_P^i$  regarding the allocated budget is a labeled set of similarity feature vectors. We build a classification model with the obtained training data that we store for a certain cluster  $C^i$ . Moreover, we maintain the similarity feature vectors  $P_{C^i}$  of the training data for each cluster to calculate similarities to new ER tasks. Considering the example in Fig. 3, we generate the corresponding models  $M_{C1}, M_{C2}$  and  $M_{C3}$  for the three clusters. Each cluster is represented by the training data that has been used for the corresponding similarity calculations.

Solving new ER problems  $p_{x,z} \in \mathcal{P}_U$  requires selecting an appropriate model for each unsolved ER problem. We propose several strategies for this selection process. The first strategy,  $sel_{base}$ , identifies the most similar cluster  $C^i$  and applies its associated model  $M_{C^i}$ . Specifically, the ER problem  $p_{x,z}$  is compared with the set of similarity feature vectors  $P_{C^i}$  from each cluster  $C^i \in \mathcal{C}_P$ . The cluster with the highest similarity score  $sim_p$  is selected, and its corresponding model  $M_{C^i}$  is used to classify the similarity vectors of  $p_{x,z}$  into matches and non-matches. In the



**Figure 3: Workflow for initializing and using an ER model repository consists of the steps: 1. Similarity Distribution Analysis, 2. ER Problem Clustering, 3. Model Generation, 4. Process new ER problems, 5. Classification.**

example, we calculate the similarities of the new ER problem  $P_{5,6}$  to the cluster representatives  $P_{C1}$ ,  $P_{C2}$  and  $P_{C3}$ .

We address potential domain shifts that may arise when the unsolved ER problem is not adequately represented by existing clusters. To handle this, we propose the selection strategy  $sel_{cov}$  described in Sect. 4.5 which integrates new ER problems and updates the existing models.

In the following section, we describe the method to compute similarities between ER problems. In detail, we provide a brief explanation of the distances used to compare two similarity distributions based on two features.

## 4.2 Similarity Distribution Analysis

Initially, we compare pairwise the existing ER tasks  $p_{k,l} \in \mathcal{P}_I$  of the data sources being initially integrated. We consider univariate and multivariate statistical analysis. For the univariate tests, we independently evaluate the distributions of each similarity feature  $f_1, \dots, f_t$ . The application of univariate distribution tests is less complex and computationally expensive compared to multivariate tests. Moreover, we assume that the feature spaces are of the same size. However, for highly heterogeneous data sources, we recommend generating record embeddings based on the available attributes for each data source and calculating similarities between these embeddings covering multiple attributes.

For the univariate tests, we calculate a distribution similarity  $sim_{dfi}$  for each feature pair and aggregate them into a weighted average similarity  $sim_p$  using the standard deviation of each feature from the ER problems.

We consider the following distribution tests to determine if the similarities of a feature  $f$  for the problems  $p_{k,l}$  and  $p_{m,n}$  are drawn from a similar distribution.

**Kolmogorow-Smirnow-Test.** This test [16] measures the maximum distance between the cumulative distribution functions  $CDF_{k,l}^f$  and  $CDF_{m,n}^f$  considering a certain feature  $f$  of the ER tasks  $p_{k,l}$  and  $p_{m,n}$ . The test statistic is defined by the supremum of the absolute difference between the two cumulative distribution functions given by Eq. (1).

$$KS = \sup_{sim} |CDF_{k,l}^f(sim) - CDF_{m,n}^f(sim)| \quad (1)$$

**Wasserstein distance.** This distance [45] measures the cost to transform a distribution  $Q$  into a distribution  $W$ . In our case, we consider the distributions  $d_{k,l}^f$  and  $d_{m,n}^f$  given by two column vectors for a feature  $f$  of the entity resolution problems  $p_{k,l}$ , resp.

$p_{m,n}$ . Using these column vectors, the corresponding cumulative distribution functions  $CDF_{k,l}^f$  and  $CDF_{m,n}^f$  are calculated and adapted to the same size if the column-wise feature vectors have different dimensions. The Wasserstein distance is the sum of the absolute differences between corresponding elements of  $CDF_{k,l}^f$  and  $CDF_{m,n}^f$ , illustrated in Eq. (2).

$$WD = \sum_{i=0}^{|CDF_{k,l}^f|} |CDF_{k,l}^f[i] - CDF_{m,n}^f[i]| \quad (2)$$

**Population stability index.** This index [40] quantifies covariate shifts between two univariate distributions  $d_{k,l}^f$  and  $d_{m,n}^f$ . Each distribution is split into bins  $B_{k,l}$  and  $B_{m,n}$  where the proportion  $prop(b_{k,l}^i)$  resp.  $prop(b_{m,n}^i)$  of similarities for the bins is determined. The index is defined as follows, where 100 is a commonly used number of bins  $|B|$ :

$$PSI = \sum_{i=0}^{|B|} (prop(b_{k,l}^i) - prop(b_{m,n}^i)) \cdot \ln \frac{prop(b_{k,l}^i)}{prop(b_{m,n}^i)} \quad (3)$$

The resulting univariate distances of the features between two ER problems  $p_{k,l}$  and  $p_{m,n}$  are transformed into similarities and averaged as  $sim_p$ . Due to the different importance of similarity features, we weigh the average by utilizing the standard deviation of a similarity feature vector to consider the discriminative power of these features.

**Classifier two-sample test.** In addition to univariate distance measures, we employ a classifier two-sample test for multivariate distribution analysis [24]. The core idea is to train a classification model to distinguish between two datasets. If the distributions are similar, the classifier's accuracy will be low, indicating that it struggles to differentiate between samples from the two datasets. We use the F1 score [4] to address the skewness according to the size of ER tasks. We define  $sim_p$  as the inverse F1 score.

## 4.3 Entity Resolution Task Clustering

We utilize the determined aggregated similarity  $sim_p$  between ER problems of  $\mathcal{P}_I$  to build an *entity resolution problem graph*  $G_p = (\mathcal{P}_I, E)$ . Each ER problem represents a vertex, and an edge  $e \in E$  represents the similarity between two ER problems. Each edge is weighted by the aggregated similarity  $sim_p$ . To prevent the formation of a single large connected component, we employ

the Leiden algorithm [43] to partition the graph into multiple clusters of similar ER problems, denoted as  $C_{\mathcal{P}}$ .

Our method is flexible and can be adapted to incorporate other graph clustering methods [48], such as label propagation [53] or the Girvan–Newman algorithm [13]. A key advantage of the Leiden algorithm is its ability to identify well-connected subgroups within weakly connected components [43]. Furthermore, the algorithm is highly scalable [43, 48], making it suitable for large graphs, which is particularly relevant when dealing with a high number of ER problems.

#### 4.4 Model Generation

We consider AL approaches and supervised ML methods for generating the corresponding model for each cluster. For the AL approaches, we distribute a given budget  $b_{tot}$  to the clusters.

**Budget distribution.** For each resulting cluster  $C^i \in \mathcal{C}_{\mathcal{P}}$ , we generate a classification model  $M_{C^i}$  to classify ER problems that are similar to those of cluster  $C^i$ . We limit the number of record pairs to build the training data by  $b_{tot}$ . We assign a minimum number of labels  $b_{min}$  to each cluster to guarantee sufficient training data. To guarantee that the total budget is sufficient, we distinguish between singleton and non-singleton clusters  $\mathcal{C}_{\mathcal{P}|s}$ , respectively.  $\mathcal{C}_{\mathcal{P}|ns}$ . If Eq. (4) holds, we have to merge the clusters of  $\mathcal{C}_{\mathcal{P}|s}$  into clusters  $C^i \in \mathcal{C}_{\mathcal{P}|ns}$  since the total budget is not sufficient for all clusters.

$$C_{\mathcal{P}} \cdot b_{min} > b_{tot} \quad (4)$$

Otherwise, the remaining budget  $b_{rem}$  determined by Eq. (5) is proportionally distributed to the clusters regarding the total number of non-singleton and singleton tasks. The ratio of the remaining budget for the non-singleton and singleton clusters is given by Eqs. (6) and (7). The budget for the singletons is distributed analogously.

$$b_{rem} = b_{tot} - b_{min} \cdot |C_{\mathcal{P}}| \quad (5)$$

$$ratio(C_{\mathcal{P}|ns}) = |C_{\mathcal{P}|ns}|/|C_{\mathcal{P}}| \quad (6)$$

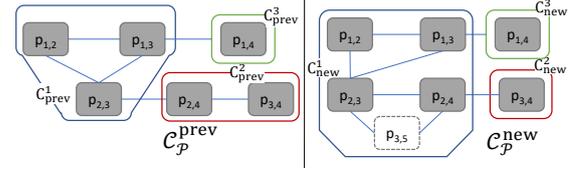
$$ratio(C_{\mathcal{P}|s}) = |C_{\mathcal{P}|s}|/|C_{\mathcal{P}}| \quad (7)$$

The budget  $b(C^i)$  for a cluster is calculated using Eqs. (8) and (9), where  $total_{C^i}$  represents the total number of feature vectors across all ER tasks within cluster  $C^i$ . If  $C^i$  is a non-singleton cluster, its budget is the sum of  $b_{min}$  and a proportion of the remaining budget  $b_{rem}$  allocated among non-singleton clusters. The proportion for each non-singleton cluster is based on its number of feature vectors relative to the total number of feature vectors in all non-singleton clusters, as defined in the denominator in Eq. (9). In contrast, if  $C^i$  is a singleton cluster, it receives a proportion of the remaining budget specifically allocated for singleton clusters, based on the total number of feature vectors from all singleton clusters.

$$C^i_{|tot} = \left| \bigcup_{p_{k,l} \in C^i} p_{k,l} \right| \quad (8)$$

$$b(C^i) = b_{min} + \frac{C^i_{|tot}}{\sum_{C^k \in \mathcal{C}_{\mathcal{P}|ns}} C^k_{|tot}} \cdot b_{rem} \cdot ratio(C_{\mathcal{P}|ns}) \quad (9)$$

**Training data selection.** A cluster  $C^i$  consists of multiple ER problems that are similar to each other. We assume that one model can classify similar ER problems. Therefore, we train one classifier for each cluster. The model generation requires training data consisting of feature vectors labeled as matches and non-matches. In our method, we integrate two AL methods, *Almser*



**Figure 4: Example of integrating a new ER problem  $p_{3,5}$ . The grey colored ER problems represent problems of  $T$ .**

[33] (see Sect. 3) and the uncertainty AL method from Mozafari et al. [27]. Note that other AL methods can be integrated. We have chosen *Almser* since it is the only method focusing on multi-source ER. Furthermore, the uncertainty method [27] scales for large ER problems and simultaneously achieves qualitative results.

**Bootstrap.** The uncertainty method from Mozafari et al. [27] utilizes a bootstrapping technique to compute an uncertainty measure. The process generates  $k$  classifiers based on the current training dataset  $T$  by sampling with repetition. The determined models  $m_1, \dots, m_k$  classify the unlabeled edge feature vectors, where the predictions are utilized to calculate the uncertainty  $unc(w)$  of a similarity feature vector  $w$  shown in Eq. (10). The term  $m_i(w)$  results in 0 or 1 if the feature vector  $w$  represents a non-match or a match, respectively.

$$unc(w) = \frac{\sum_{i=1}^k m_i(w)}{k} \cdot \left( 1 - \frac{\sum_{i=1}^k m_i(w)}{k} \right) \quad (10)$$

As an extension, we introduce a score  $s$  for each similarity feature vector  $w$  representing how unique a feature vector is regarding the associated records. The calculation of the score is similar to the inverse document frequency (IDF), considering the related records as words and the cluster as documents. We compute a record score  $s_r$  and the derived score  $s$  for a similarity vector  $w$  as shown in Eqs. (11) and (12). For each record, we determine the number of clusters where the record  $r$  occurs  $|C_{\mathcal{P}|r}|$  and the total number of clusters. The score  $s$  of a similarity feature vector  $w$  is the average over the adjacent records  $src$  and  $tgt$  of the feature vector.

$$s(w) = [s_r(src(w)) + s_r(tgt(w))] / 2 \quad (11)$$

$$s_r(r) = \log \frac{|C_{\mathcal{P}}|}{|C_{\mathcal{P}|r}|} \quad (12)$$

**Almser.** We use the original implementation<sup>1</sup> and extend it to support batch processing. As input, we only consider the induced subgraph regarding the ER tasks of a cluster.

In addition to the model generation, we maintain the selected vectors by the AL method as the set  $P_{C^i}$  for each cluster  $C^i$ . We utilize the sets to determine the best cluster for a new data source and the associated ER problems.

#### 4.5 Solving New Entity Resolution Problems

Integrating a new data source requires solving the ER problem  $p_{x,z} \in \mathcal{P}_{\mathcal{U}}$ . We consider different strategies to determine an appropriate model. The first strategy,  $sel_{base}$ , assumes minimal domain shift, eliminating the need for new models or integration of additional ER problems. Here, the ER problem  $p_{x,z}$  is compared with the set of feature vectors  $P_{C^i}$  used to train models for each cluster  $C^i$ . We employ the same distribution test used to construct the ER problem graph  $G_{\mathcal{P}}$ , incorporating the standard deviation

<sup>1</sup><https://github.com/wbgs-uni-mannheim/ALMSER-GB>

of similarity vectors to weigh similarities. The cluster  $C^i$  with the highest similarity  $sim_p$  to  $p_{x,z}$  is selected, and the corresponding model  $M_{C^i}$  is applied to the feature vectors  $w \in P_{x,z}$ .

The second strategy  $sel_{cov}$  addresses domain shifts in similarity feature distributions. For these methods, the new ER problem  $p_{x,z} \in \mathcal{U}$  is compared with all ER problems in  $\mathcal{P}_I$ . We update the ER problem similarity graph  $G_P$  by adding the ER problem  $p_{x,z}$  and connecting it with the existing problems  $p_{k,l} \in \mathcal{P}_I$ . The graph is then reclustered using the Leiden algorithm already used before. We distinguish between two sets of ER problems:  $T$ , which includes problems already used for selecting training data, and  $U$ , which includes problems not yet used. A new model is trained if the unsolved problem  $p_{x,z} \in C$  belongs to a cluster where all ER problems  $p_{k,l} \in C$  are in  $U$ . Otherwise, the model  $M_{C_{prev}}$  from the previous cluster  $C_{prev} \in \mathcal{C}_P^{prev}$  is reused. Reclustering may result in new clusters  $C^{new}$  where ER problems from different previous clusters are grouped. For example, as shown in Fig. 4,  $p_{2,4} \in C_{prev}^2$  from the previous cluster is reassigned to  $C_{new}^1$  in the new clusters  $\mathcal{C}_P^{new}$ . In such cases, the model  $M_{C_{prev}}$  from the cluster  $C_{prev}$  with the maximum overlap according to  $C_{new}$  is selected.

As the number of new similarity vectors increases, it becomes more likely that the associated model  $M_C$  for a cluster  $C$  will potentially lead to the underrepresentation of the new ER problems within the cluster. Therefore, we define a coverage ratio  $cov$  where we update the model  $M_{C_{prev}}$  if  $cov$  exceeds a predefined threshold  $t_{cov}$ . We compute the coverage ratio as the proportion of similarity feature vectors from ER problems in  $U$  relative to all similarity feature vectors in cluster  $C^{new}$ .

$$cov(C) = \frac{\{w \mid w \in U \cap C\}}{\{w \mid w \in C\}} \quad (13)$$

Updating a model requires new training data. Therefore, we use similarity feature vectors from ER problems  $p_{k,l} \in U \cap C^{new}$ . We use the same AL method as for  $\mathcal{P}_I$  (see Sect. 4.4) with the following total budget to select the training data:

$$b_{new} = b_{tot} \cdot cov(C) \cdot |\{w \mid w \in T \cap C^{prev}\}| / b_{tot} \quad (14)$$

We derive the budget from the originally used one  $b_{tot}$ , the coverage ratio  $cov(C)$  defined above, and the ratio of record pairs used to train the previous model  $M_{C_{prev}}$  to the total training budget. After updating the model, the set of ER problems used for training  $T$  is extended and is removed from  $U$ . The updated graph  $G_P$  and new clusters  $\mathcal{C}_P^{new}$  are then used for new problems of  $\mathcal{P}_U$ .

## 5 Evaluation

In this section, we evaluate our method *MoRER* using three datasets, considering the multi-source AL method *Almser* [33], the transfer learning method *TransER* [18] focusing on relational ER tasks, the self-supervised approach *Sudowoodo* [47], the SOTA supervised ER approach *Ditto* [22], and the small language model-based method *AnyMatch* [51]. Moreover, we analyze the components of *MoRER* such as the various distribution tests and the impact of retraining.

### 5.1 Datasets

Our method focuses on multi-source ER approaches, where the availability of datasets is limited. We consider three datasets being widely used in this domain [7, 21, 33, 37, 50] as summarized in Table 2.

*Dexter*. This dataset is derived from the camera dataset of the ACM SIGMOD 2020 Programming Contest. The dataset consists

**Table 2: Statistics of the datasets used in our evaluation.**

Name	# ER problems	# Record pairs	# Matches
<u>D</u> xter	276	1,100K	368K
<u>W</u> DC-computer	12	74.5K	4.8K
<u>M</u> usic	20	385.9K	16.2K

of 23 sources with around 21,000 records and intra-source duplicates. Each data source consists of source-specific attributes.

*WDC-computer*. We utilize the WDC training corpus for large-scale product matching and use the same subset derived from the original one being used in the *Almser* study [33], consisting of four data sources.

*Music*. The MusicBrainz dataset is a synthetically generated dataset from the MusicBrainz (<https://musicbrainz.org/>) database. The dataset was corrupted by [15] and it consists of five sources with duplicates for 50% of the original records. Each data source is duplicate-free, but the records are heterogeneous regarding the characteristics of attribute values, such as the number of missing values, the length of values, and the ratio of errors.

### 5.2 Experimental Setup

We implemented *MoRER* in Python 3.12 using scikit-learn 1.5.1 and networkx 3.3. We modified the existing *Almser* implementation as available on GitHub and reimplemented the uncertainty active learning method from Mozafari et. al. [27], which we called *Bootstrap*. We set  $k=100$  achieving qualitative results following previous studies [5, 27]. We evaluate the effectiveness by calculating precision, recall, and F1 score [4] according to the predicted matches across overall ER tasks for a test dataset.

All experiments were performed on a computer with a 64-core AMD(R) EPYC(R) 7713 2.0GHz - Turbo 3.7GHz, 50GB allocated RAM, and an Nvidia Tesla A30. In the following, we will describe the generation of similarity feature vectors and the considered parameters of *MoRER*.

**Similarity Feature Vectors.** The main focus of our method is the classification and, consequently, the model generation and selection. Therefore, we pre-calculate the similarity vectors of the ER problems. We use the same attribute comparisons, string similarity functions, and blocking keys for the *Dexter* dataset as in previous studies [21, 36]. For the *WDC-computer* and *Music* datasets, we utilize the feature vectors<sup>2</sup> used in the previous work presenting *Almser* [33]. The authors use various string similarity functions for textual data and normalized differences for numerical values.

**Method Parameters.** We use the following configuration summarized in Table 3, where the values written in bold are default values. We split the available ER problems into 50% initial tasks  $\mathcal{P}_I$  and consequently 50% unsolved ER tasks  $\mathcal{P}_U$  for the *Dexter* dataset. Note that we also consider the linkage between the same data sources since the data sources are not duplicate-free for the *Dexter* dataset. For the *WDC-computer* and *Music* datasets, we utilize the train-test split of the provided feature vectors. If a data source pair  $(D_1, D_2)$  consists of record pairs  $(r_k, r_l) \in train$  and  $(r_m, r_n) \in test$  being present in the training and test dataset, we built two ER problems with data source pairs  $(D_{1train}, D_{2train})$  and  $(D_{1test}, D_{2test})$ .

<sup>2</sup>[http://data.dws.informatik.uni-mannheim.de/benchmarkmatchingtasks/almser\\_gen\\_data/](http://data.dws.informatik.uni-mannheim.de/benchmarkmatchingtasks/almser_gen_data/)

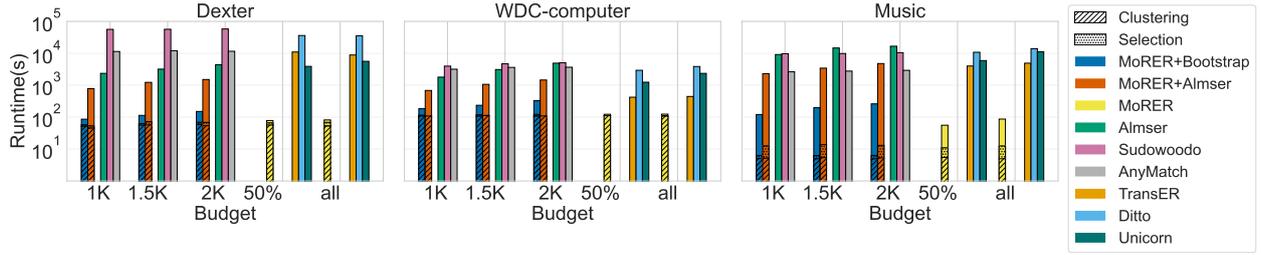


Figure 5: Runtime comparison of *MoRER* with AL and the supervised version to *Almser*, *Sudowoodo*, *AnyMatch*, *TransER*, and *Ditto*. The shaded areas represent the runtime for the statistical analysis/clustering (striped) and the selection (dotted) of a model.

Table 3: Overview of the parameter setting for *MoRER*. The values in bold represent the default values.

Name	Values
$ratio_{init}$	<b>50%</b> , 30%
Distribution test	<b>KS</b> , WD, PSI, C2ST
Model generation	<b>AL</b> , supervised
AL method	Bootstrap [27], <i>Almser</i> [33]
Selection method	<b>sel<sub>base</sub></b> , <i>sel<sub>cov</sub></i>

**Compared methods.** Due to the limitations of existing methods for reusing ML models, we focused on label-efficient techniques like active learning and transfer learning. Our method is compared to *Almser* [33], and *TransER* [18]. All methods utilized identical similarity feature vectors, with a consistent division between solved and unsolved ER tasks to ensure comparability.

We also compare our method against language model-based approaches, which have shown superior performance over traditional methods for processing textual data. To generate record embeddings, we utilize the same set of attributes originally employed to create the similarity vectors. We consider *Sudowoodo* [47], *Ditto* [22], *Unicorn* [11], and *AnyMatch* [51].

We utilize the same record pairs for the training and test sets for each method. We partition the training and test data sets by the source pairs of the corresponding record pairs to generate ER problems for  $\mathcal{P}_I$  and  $\mathcal{P}_U$ . Unlike AL methods (such as *Almser*), *TransER*, *Ditto*, and *Unicorn* are not designed to select informative links. Instead, we treat all initial ER tasks  $\mathcal{P}_I$  among the integrated data sources as training data and 50% of the similarity feature vectors for training. This consideration results in the following training data sizes of all pairs for the *Dexter*, *WDC-computer*, and *Music* datasets: 68,339, 231,521, and 46,049 record pairs, respectively. These training sizes reflect the volume of labeled record pairs used by *TransER*, *Ditto*, and *Unicorn* for their classification tasks. For *Sudowoodo* and *AnyMatch*, we use the semi-supervised approach with the same labeling budget as our method *MoRER* and *Almser*.

*Almser* [33]. We utilize the integrated data sources and their corresponding ER tasks, maintaining the same number of labeled pairs as our method. To ensure fairness, we modified the original implementation<sup>3</sup> to support batch processing and use graph-inferred labels from cleaned connected components as per the original study.

*TransER* [18]. We use the implementation available on GitHub<sup>4</sup>. *TransER*'s parameter values are set at [18]  $k = 10$ ,  $t_c = 0.9$ ,  $t_l = 0.9$ , and  $t_p = 0.9$ , as determined through pre-experiments. In this context,  $k$  represents the neighborhood size of a feature vector, while  $t_c$  and  $t_l$  are the thresholds for class confidence similarity and structural similarity between the neighborhoods of the source and target domains. Additionally,  $t_p$  denotes the threshold for the confidence of pseudo labels.

*Sudowoodo* [47]. We use the implementation on GitHub<sup>5</sup> with the default parameters presented in the original work [47]. Moreover, we consider all proposed optimizations showing the best results according to the different datasets. For comparability, we use the semi-supervised variant with the same labeling budget.

*Ditto* [22] and *Unicorn* [11]. For faster training, we utilize DistilBERT [38] as a pre-trained model. Moreover, for *Ditto*, we use a maximum sequence length of 256, also fixed in the original work [22], and train the model for 10 epochs. We report the results that lead to the smallest training loss over the 10 epochs. For *Unicorn*, we use the default parameter values of the implementation on GitHub<sup>6</sup> with six experts. We utilize the combined loss function covering the diversity and a balanced importance of experts.

We evaluate *AnyMatch* [51] using its official GitHub implementation<sup>7</sup>. For each dataset, the model is trained on the corresponding training set and evaluated on the test set, with the parameterized sample size  $n_r$  for comparability. Note, *AnyMatch* filters relevant record pairs using the full ground truth of the training data. We use a batch size of 32, GPT-2 as the language model, and a learning rate of  $2^{-5}$ .

### 5.3 Comparison to Baselines

**Effectiveness.** The results, shown in Table 4, indicate that *MoRER* performs comparably to *Almser* under certain budgets and active learning configurations. *MoRER*, when combined with *Almser*, outperformed both the *Bootstrap* method and *Almser* alone on the *Dexter* dataset. We hypothesize that *Almser* benefits from the high number of entity resolution tasks, resulting in rich similarity graphs, even when these graphs are divided due to the separation of ER tasks. Moreover, the partitioning of ER tasks reduces the search space for identifying informative links for similar classes of entity resolution tasks, thereby minimizing the likelihood of selecting uninformative feature vectors.

<sup>4</sup><https://github.com/nishadi/TransER>

<sup>5</sup><https://github.com/megagonlabs/sudowoodo>

<sup>6</sup><https://github.com/ruc-datalab/Unicorn>

<sup>7</sup><https://github.com/jantory/anymatch>

<sup>3</sup><https://github.com/wbgs-uni-mannheim/ALMSER-GB>

**Table 4: Linkage quality (Precision/Recall/F1 score) comparison of the budget limited methods *MoRER* in combination with an AL method to *Almser* standalone, *Sudowoodo*, and *AnyMatch*, and the supervised methods *MoRER*, *TransER*, and *Ditto*. We highlight the best F1 score in bold and underline the second best one for each budget and method type (AL vs supervised).**

D B	MoRER+Almser	MoRER+BS	Almser	Sudowoodo	AnyMatch	B	MoRER	Ditto	Unicorn	TransER
1K	0.96/0.92/ <b>0.94</b>	0.88/0.90/0.89	0.97/0.87/0.92	0.42/0.76/0.54	0.59/0.96/0.73	50%	0.94/0.92/ <b>0.93</b>	0.60/0.99/0.75	0.53/0.96/0.68	0.95/0.90/0.92
D 1.5K	0.96/0.92/ <b>0.94</b>	0.85/0.94/0.89	0.97/0.89/0.92	0.43/0.68/0.53	0.60/0.97/0.74	all	0.91/0.93/ <b>0.92</b>	0.60/0.99/0.75	0.56/0.93/0.70	0.94/0.87/0.90
2K	0.96/0.90/0.93	0.88/0.93/0.90	0.97/0.90/0.94	0.45/0.85/0.58	0.59/0.98/0.73					
1K	0.84/0.83/0.83	0.85/0.77/0.81	0.93/0.92/0.93	0.43/0.99/0.60	0.55/0.97/0.70	50%	0.94/0.93/0.94	0.92/0.95/0.93	0.77/0.92/0.84	0.79/0.86/0.82
W 1.5K	0.94/0.87/0.90	0.90/0.86/0.88	0.94/0.93/0.93	0.40/0.99/0.57	0.58/0.94/0.72	all	0.94/0.92/0.93	0.98/0.94/0.96	0.90/0.90/0.90	0.80/0.86/0.83
2K	0.95/0.94/0.94	0.79/0.84/0.84	0.95/0.93/0.94	0.47/0.90/0.61	0.61/0.98/0.75					
1K	0.98/0.96/0.97	0.94/0.92/0.97	0.98/0.97/0.98	0.92/0.95/0.94	0.97/0.99/0.98	50%	0.99/0.96/0.97	0.99/0.99/0.99	0.98/0.98/0.98	0.96/0.93/0.94
M 1.5K	0.98/0.97/0.98	0.98/0.97/0.97	0.99/0.97/0.98	0.95/0.96/0.96	0.93/0.99/0.96	all	0.99/0.97/0.98	1.00/0.99/0.99	0.97/0.99/0.98	0.95/0.92/0.94
2K	0.99/0.96/0.98	0.96/0.96/0.97	0.99/0.97/0.98	0.93/0.96/0.95	0.96/0.99/0.97					

*Almser* outperforms *MoRER+Bootstrap* by up to 9% for budgets of 1000 and 1500. For the *Dexter* dataset, *MoRER*'s average F1 score using the Bootstrapping-based AL method was approximately 4.3% lower than *Almser*'s. For the *Music* dataset, all *MoRER* combinations and *Almser* achieve similar results, with F1 score differences of at most 0.01 for a budget above 1500.

In contrast, *TransER* consistently underperformed relative to the combined performance of *Almser* and *MoRER* across all tasks, regardless of whether the full training dataset or the 50% version was utilized. This consistent underperformance shows its limitations in handling large volumes of heterogeneous training data, which is particularly evident in the *Dexter* and *WDC-computer* datasets. Considering the language model-based methods without using labeled training data, *Sudowoodo* and *MultiEM* achieve comparable results compared to the aforementioned methods according to the *Music* dataset. However, the approaches result in a decreased F1 score of up to 38% (*Dexter*) when comparing *Sudowoodo* with *MoRER+Almser*. *Sudowoodo* was originally conceptualized for pairwise record linkage. Therefore, we assume that the various and heterogeneous ER problems in *Dexter* and *WDC-computer* cause the challenge of determining representations to distinguish matches from non-matches.

*AnyMatch* and *Unicorn* achieve comparable performance only on the *Music* dataset. For the *WDC-computer* dataset, the results of *AnyMatch* decrease by up to 22% compared to *MoRER+Almser* and by 17.4% when using *MoRER* with *Bootstrap*. We assume this decrease is caused by the limitations of the AutoML selection, which appears insufficient for building effective models when classifying large numbers of record pairs. For *Dexter* and *WDC-computer*, *Unicorn* generates results that decreased by up to 25% in F1 score compared to *MoRER+supervised*.

*Ditto*, when utilizing the complete training datasets, outperforms the *MoRER+Almser* method by an average of 3% and the *Bootstrap* combination by up to 6% for budgets exceeding 1500. However, for the *Dexter* dataset, the *MoRER* combinations achieved higher F1 scores than *Ditto*. The *Dexter* dataset, comprising camera specifications with numerous model numbers, introduces challenges due to minor textual differences that can lead to non-matches. We hypothesize that pretrained language model-based methods, such as *Ditto*, struggle with accurately classifying matches and non-matches when only small textual distinctions exist, particularly in scenarios with limited training data. We can also observe the importance of the training data size, considering the results achieved with 50% of the training data. The difference in F1 score is reduced, or *MoRER* achieves higher F1 scores, as observed in the *WDC-computer* dataset with a budget of 2000.

**Efficiency.** We consider the total runtime that encompasses training data generation, model training, and classification. Notably, for AL methods, the time required to select training data significantly impacts the overall labeling process, since the selection of informative samples often is complex and computationally intensive. *MoRER* combined with the *Bootstrap* AL method demonstrates significant advantages, outperforming both *Almser* and *TransER* across all datasets and labeling budgets, as shown in Fig. 5. The additional effort required for statistical analysis, ER clustering and model selection is minimal, ranging from 6 seconds (*Music*) to 121 seconds (*WDC-computer*). Compared to selecting training data using an AL method, the proportion ranges from 0.3% (*Music* with *Almser*) to 67% (*Dexter* with *Bootstrap*).

For example, on the *Music* dataset, *MoRER+Bootstrap* runtimes range from 121 to 2621 seconds, whereas *Almser* can take up to 4.6 hours. *Almser*'s longer runtimes are due to the increasing size of similarity graphs, which require more computational resources for graph analysis. The efficiency advantage results from the *MoRER* strategy of clustering similar ER tasks among the integrated data sources to select training data. This clustering approach significantly decreases the search space for potential links, contributing to the observed performance improvements. Reducing the search space also positively influences the combination of *MoRER* and *Almser*. The performance boost results in a maximum speedup of 4.3x (*Music* dataset, budget = 1000) over *Almser* alone. *TransER*'s runtimes range from approximately 7 minutes (*WDC-computer*) to 2.5 hours (*Dexter*), considering the complete training dataset and 50% of it.

For datasets consisting of many feature vectors, the clustering approach in *MoRER* reduces the search space for selecting training data. In contrast to *MoRER*, *TransER* compares each unsolved feature vector with all feature vectors from the integrated ER tasks. *Ditto* is slower than *MoRER* combined with *Almser* or *Bootstrap* due to its training times. In contrast, *AnyMatch* and *Unicorn* achieve smaller runtimes by using smaller training datasets resp. less complex models. However, neural network-based methods, including *Ditto*, *Unicorn*, and *AnyMatch*, generally require more training time than non-language model-based approaches such as *MoRER*, *Almser*, and *TransER*. Notably, *Sudowoodo* exhibits the longest runtimes, reaching up to 16 hours on the *Dexter* dataset.

Table 5 presents a summary of the speedup factors for *MoRER* in comparison to other supervised ML methods. The analysis highlights that using *Bootstrap* as an AL method within *MoRER* results in significant runtime improvements over *Almser*, especially when compared to *Almser* alone, *TransER*, *Ditto*, and *Sudowoodo*. In terms of effectiveness, *MoRER* combined with the *Bootstrap*

**Table 5: Summary of speedup factors for *MoRER* with various AL methods compared to the approaches (*Alm*)*ser*, *TransER* (TER), (*Su*)*dowoodo*, (*Dit*)*to*, (*Uni*)*corn*, and (*Any*)*Match* regarding the datasets (*Dex*)*ter*, (*Mus*)*ic*, and (*WDC*)*-computer*.**

DS	Budget: 1000									Budget: 1500									Budget: 2000								
	Alm	TER	TER	Su	Dit	Dit	Uni	Uni	Any	Alm	TER	TER	Su	Dit	Dit	Uni	Uni	Any	Alm	TER	TER	Su	Dit	Dit	Uni	Uni	Any
	50%	all	all	50%	all	50%	all	50%	all	50%	all	50%	all	50%	all	50%	all	50%	all	50%	all	50%	all	50%	all	50%	all
<i>MoRER+Almser</i>																											
Dex	3.0	14.1	11.3	72	45.7	45.3	5	7.2	14.6	2.6	8.9	7.2	46	29.0	28.8	3.2	4.6	9.8	2.9	7.3	5.9	38.3	23.6	23.4	2.6	3.7	7.8
Mus	4.0	1.7	2.1	4.2	4.6	6.0	2.6	4.9	1.1	4.3	1.2	1.4	2.9	3.1	4.1	1.7	3.3	0.8	3.5	0.8	1.0	2.2	2.2	2.9	1.2	2.3	0.6
WDC	2.6	0.6	0.6	5.8	4.3	5.6	1.8	3.4	4.6	2.9	0.4	0.4	4.4	2.7	3.6	1.2	2.2	3.4	3.4	0.3	0.3	3.5	2.0	2.6	0.9	1.6	2.5
<i>MoRER+Bootstrap</i>																											
Dex	27.3	127.3	102.7	46	413.8	410.5	45.1	65	132.2	28.1	96.6	78.0	41.2	314.1	311.6	34.3	49.4	106.1	28.9	72.8	58.7	33.4	236.5	234.6	25.8	37.2	76.8
Mus	75.7	33.1	40.6	80.6	87.4	115.4	49	92.8	74.3	21.9	20.2	24.8	49.9	53.5	70.6	30	56.7	14	63.8	15.3	18.8	40	40.4	53.4	22.7	42.9	11.2
WDC	9.8	2.3	2.4	306	16.0	21.0	6.8	12.8	17.3	13.2	1.8	1.9	240.5	12.5	16.4	5.3	10	15.4	15.0	1.3	1.4	176.7	8.9	11.8	3.8	7.2	11.2

AL method leads to a decrease in quality compared to the *Almser* combination. Due to the missing training data generation step, the supervised-based *MoRER* achieves the best speedup compared to the other approaches.

#### 5.4 Analysis of Distribution Tests

This section compares the various distribution tests for computing the similarity between ER problems. The similarity is essential for grouping tasks and determining which model  $M_C^i$  of a cluster  $C_p^i$  is used for a new ER problem  $D_l \in \mathcal{N}$ . Fig. 6 presents the results of the three distribution tests: Kolmogorov-Smirnov (KS), Wasserstein distance (WD), Population Stability Index (PSI), and the classifier two-sample test (C2ST).

For the homogeneous *Music* dataset, all distribution tests performed similarly across both AL methods, indicating that the choice of distribution test is less critical for datasets with consistent characteristics. However, for heterogeneous or noisy datasets like *Dexter* and *WDC-computer*, the choice of distribution test and its combination with specific AL methods significantly impacts performance.

For example, the Wasserstein Distance achieved strong results for the *WDC-computer* dataset when paired with the *Bootstrap* method, but its performance was inconsistent for the *Dexter* dataset under the same method. This variability highlights the sensitivity of WD to dataset characteristics and AL configurations. In contrast, PSI demonstrated more robust and consistent results across different AL methods and datasets, making it a reliable choice for scenarios with diverse or noisy data. Notably, PSI achieved its best results for specific configurations, such as the *Dexter* dataset with a budget of 1000.

The Kolmogorov-Smirnov test and the multivariate analysis classifier two-sample test, on the other hand, exhibited strong overall performance and proved to be reliable options across various datasets and AL methods. It slightly outperformed the other tests for the *Dexter* dataset, demonstrating its effectiveness in handling heterogeneous data.

#### 5.5 Analysis of Selection Methods

We evaluate the impact of various selection methods for determining an appropriate model for a new ER problem, as described in Sect. 4.5. We use an initial budget of 1000 and the *Bootstrap* AL method. The first method, denoted as  $sel_{base}$ , assigns an ER problem to the most similar cluster without any retraining. The second method,  $sel_{cov}$ , integrates the new ER problem into the ER problem graph  $G_p$  and may update the existing models.

We analyze the ratios of uncovered similarity feature vectors during the training data generation step, specifically  $t_{cov} = [0.1, 0.25, 0.5]$ . For example, a ratio of 0.1 indicates that retraining with new training data occurs if a cluster contains at least 10% of similarity feature vectors that were not included in the previous training data generation steps. Fig. 7 illustrates the results comparing the base selection strategy with the reclustering strategy in terms of resulting F1 scores and the increased labeling budget.

The reclustering selection strategy leads to improved ER quality compared to the base selection strategy. The  $sel_{cov}$  achieves the best results for the *WDC-computer* and *Music* datasets, utilizing a threshold  $t_{cov}$  of 0.1. However, these strategies also require the most additional labeling effort compared to the other selection methods. In the *Dexter* dataset, we observe a negative impact from a small covered ratio, such as 0.1. Due to the wide range of sizes of ER problems in the *Dexter* dataset, a single ER problem can trigger unnecessary retraining, as indicated by higher F1 scores when using a threshold  $t_{cov}$  greater than 0.1.

## 6 Discussion

We introduce a novel method for reusing and creating classification models for integrated data sources under a limited labeling budget. This section discusses research questions, including limitations and suggestions regarding configurations and alternatives.

**RQ1: When should our method be applied?** Our method performs best if more data sources are integrated, so a distinction between various ER problems and different classification models makes sense. For a small number of ER problems of moderate size, the resulting tasks can be solved using a single model determined by a sophisticated AL method such as *Almser*. However, due to the increasing runtime regarding a high number of integrated data sources, our method is more efficient and reaches similar or higher F1 scores compared to *Almser*. Transfer learning approaches are useful when it is known which source domain can be adapted to a certain target domain. Otherwise, the size of the training data from the source domain must be very large, as a small amount is not sufficient for transferring. When ER problems originate from a general domain and sufficient training data is available or can be generated, *Ditto* yields qualitative results, particularly when leveraging data sources rich in textual information.

**RQ2: What are the additional costs of our method?** To reuse models for new ER tasks, we initially compute pairwise the similarities between the available ER tasks and cluster the

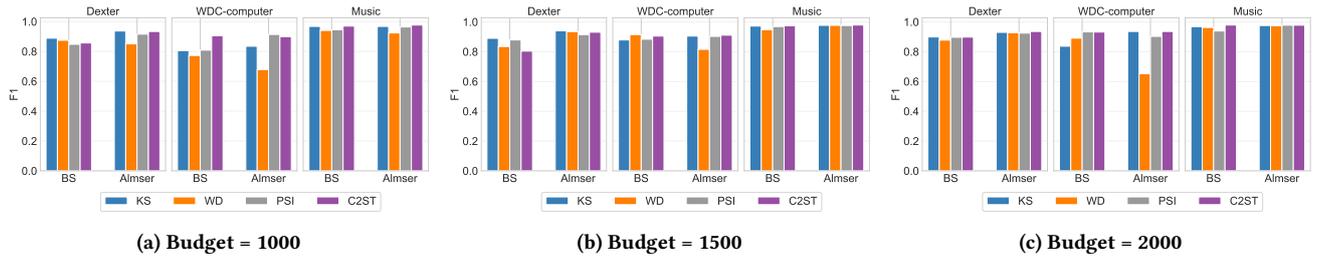


Figure 6: Comparison of the distribution tests using the AL methods *Bootstrap* (BS) and *Almser*.

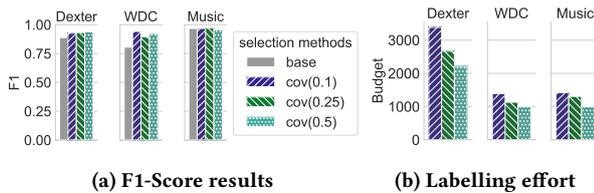


Figure 7: Comparison of the selection strategies  $sel_{base}$  and  $sel_{cov}$  using Bootstrap AL ( $b=1000$ ).

resulting ER problem similarity graph. The similarity computation and the clustering scale with the number of initial ER tasks. However, the proportion of the total runtime is below 1% with *Almser* and 70% using *Bootstrap*. Moreover, due to the reduced sets of record pairs for each cluster, the selection of informative record pairs using an AL method is more efficient than the application on the union of all ER tasks. In a real-world application, we would apply the retraining of clusters described in Sect. 4.5 due to potential domain shifts. Consequently, we would run the clustering step for each new ER tasks, which is a moderate effort due to the efficiency of the Leiden algorithm [43]. Moreover, we would retrain the underlying models with new selected training data from the upcoming ER tasks.

**RQ3: Which active learning approach should be used?** As the results indicate, our modified Bootstrapping outperforms the baseline methods and the combination with *Almser* regarding efficiency and achieves comparable results for clean ER problems. However, for dirty ER problems where data sources with duplicates are involved and a high number of problems, the combination with *Almser* should be used. *Almser* benefits from the reduced search space of candidate record pairs, allowing *Almser* to focus on the most informative instances for labeling. The combination can be applied to heterogeneous and noisy datasets. If runtime efficiency is a critical requirement and the datasets are cleaner (e.g., *Music* dataset), the *Bootstrap* AL method can be considered.

**RQ4: Which distribution test for computing the similarity between ER problems is the most effective?** The Wasserstein distance results in the lowest F1 score for the bootstrapping-based AL approach on the *Dexter* and *Music* datasets. We hypothesize that the Wasserstein distance generates overly similar results for these datasets, which negatively affects performance, particularly in datasets with many ER tasks (*Dexter*) or a large number of feature vectors (*Music*). However, the Wasserstein distance outperforms the Kolmogorov-Smirnov test for the impure and smaller *WDC-computer* dataset, utilizing a certain

budget. Meanwhile, the Population Stability Index achieves moderate linkage quality across the datasets for the *Bootstrap* AL method.

We conclude that for homogeneous datasets like *Music*, the selection of a certain distribution test is less crucial than for more heterogeneous or noisy datasets like *Dexter* and *WDC-computer*. Moreover, the results suggest that *Almser* benefits more from specific distribution tests (e.g., Kolmogorov-Smirnov for *Dexter*), whereas *Bootstrap* shows less variability across tests.

**RQ5: What are the limitations of our method?** Our method assumes the availability of a sufficient number of solved ER problems. However, when the number of solved problems is limited, the diversity of feature vectors becomes constrained. This limitation can result in models that fail to generalize effectively across the broader spectrum of ER problems, leading to a significant domain shift between solved and unsolved problems.

To address this issue, we proposed selection strategies integrating the new ER problems in combination with an updating mechanism of the existing models. This process ensures that the models remain adaptable to evolving data distributions. In this work, we only consider the heterogeneity regarding the feature space. Therefore, we can only apply our method to ER problems with common features. A trivial strategy to handle data sources with different attributes is to apply a pre-trained language model to determine a comparable representation.

## 7 Conclusion

The classification step in ER is crucial. The majority of existing approaches use ML models that require training data. Existing methods do not address the challenges regarding multiple classification tasks occurring in multi-source ER scenarios due to the increasing effort to generate training data. Our method addresses these issues by creating a model repository to reuse pre-trained classifiers for new ER problems. For initializing the repository, we cluster similar ER problems and derive one model for each group. We propose different variants to determine an appropriate model. In the base variant, we apply the most suitable built model of a similar cluster for a new ER task. The second selection method addresses potential domain shifts by updating the clusters and the models depending on the ER problems of a cluster.

The evaluation demonstrates comparable quality to *Almser* and superior results to *TransER*, with significant runtime improvements, highlighting the method’s scalability and practical value. Compared to the language model-based approaches, *MoRER* achieves comparable performance or outperforms *Ditto* using 50% of the training data, ranging from roughly 23K to 115K labeled record pairs. Comparing *MoRER* with the language model-based approaches *AnyMatch*, *Unicorn*, and *Sudowoodo*, our method significantly outperforms them.

In future work, we aim to integrate our method into a comprehensive system for storing and querying solved ER problems, clusters, and classification models, enabling solution reuse for new tasks. While our approach assumes a shared feature space, we suggested the combination of attributes into a single embedding using pre-trained language models. We also plan to explore incrementally expanding the feature space for specific ER tasks. The effectiveness of the reuse depends on the clusters based on the ER problem graph and the similarities. As an additional criterion for the model performance of a cluster, we will investigate the relationship between model performance and cluster stability measures.

## Artifacts

The reference code is available from our repository at [https://github.com/vicolinho/Model\\_Reuse\\_ER](https://github.com/vicolinho/Model_Reuse_ER).

## Acknowledgments

The authors acknowledge the financial support by the Federal Ministry of Education and Research of Germany and by Sächsische Staatsministerium für Wissenschaft, Kultur und Tourismus in the programme Center of Excellence for AI-research "Center for Scalable Data Analytics and Artificial Intelligence Dresden/Leipzig", project identification number: ScaDS.AI.

This work was partially funded by Universities Australia and the German Academic Exchange Service (DAAD) grant 57701258.

## References

- [1] 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, Washington, DC, USA, August 24 - 27, 2003*. ACM, 39–48. doi:10.1145/956750.956759
- [2] Ursin Brunner and Kurt Stockinger. 2020. Entity Matching with Transformer Architectures - A Step Forward in Data Integration. In *Proceedings of the 23rd International Conference on Extending Database Technology, EDBT, OpenProceedings.org*, 463–473. doi:10.5441/002/EDBT.2020.58
- [3] Peter Christen. 2012. *Data Matching – Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Springer. doi:10.1007/978-3-642-31164-2
- [4] Peter Christen, David J. Hand, and Nishadi Kirielle. 2023. A Review of the F-Measure: Its History, Properties, Criticism, and Alternatives. *Computing Surveys* 56, 3 (2023), 1–24.
- [5] Victor Christen, Peter Christen, and Erhard Rahm. 2019. Informativeness-Based Active Learning for Entity Resolution. In *Machine Learning and Knowledge Discovery in Databases - International Workshops of ECML PKDD (Communications in Computer and Information Science, Vol. 1168)*. Springer, 125–141. doi:10.1007/978-3-030-43887-6\_11
- [6] Vassilis Christophides, Vasilis Efthymiou, Themis Palpanas, George Papadakis, and Kostas Stefanidis. 2021. An Overview of End-to-End Entity Resolution for Big Data. *ACM Comput. Surv.* 53, 6 (2021), 127:1–127:42. doi:10.1145/3418896
- [7] Fernando de Meer Pardo, Claude Lehmann, Dennis Gehrig, Andrea Nagy, Stefano Nicoli, Branka Hadji Misheva, Martin Braschler, and Kurt Stockinger. 2025. GraLMatch: Matching Groups of Entities with Graphs and Language Models. In *Proceedings 28th International Conference on Extending Database Technology, EDBT, OpenProceedings.org*, 1–12. doi:10.48786/EDBT.2025.01
- [8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT*. Association for Computational Linguistics, 4171–4186. doi:10.18653/v1/N19-1423
- [9] AnHai Doan, Pradap Konda, Paul Suganthan G. C., 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. doi:10.1145/3405476
- [10] Muhammad Ebraheem, Saravanan Thirumuruganathan, Shafiq R. Joty, Mourad Ouzzani, and Nan Tang. 2017. DeepER - Deep Entity Resolution. *CoRR abs/1710.00597* (2017). arXiv:1710.00597 <http://arxiv.org/abs/1710.00597>
- [11] Ju Fan, Jianhong Tu, Guoliang Li, Peng Wang, Xiaoyong Du, Xiaofeng Jia, Song Gao, and Nan Tang. 2024. Unicorn: A Unified Multi-Tasking Matching Model. *SIGMOD Rec.* 53, 1 (2024), 44–53. doi:10.1145/3665252.3665263
- [12] Leonardo Gazzarri and Melanie Herschel. 2023. Progressive Entity Resolution over Incremental Data.. In *EDBT*. 80–91.
- [13] Michelle Girvan and Mark E. J. Newman. 2002. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences* 99, 12 (2002), 7821–7826. doi:10.1073/pnas.122653799
- [14] Rihan Hai, Christos Koutras, Christoph Quix, and Matthias Jarke. 2023. Data Lakes: A Survey of Functions and Systems. *IEEE Trans. Knowl. Data Eng.* 35, 12 (2023), 12571–12590. doi:10.1109/TKDE.2023.3270101
- [15] Kai Hildebrandt, Fabian Panse, Niklas Wilcke, and Norbert Ritter. 2020. Large-Scale Data Pollution with Apache Spark. *IEEE Trans. Big Data* 6, 2 (2020), 396–411. doi:10.1109/TBDATA.2016.2637378
- [16] Myles Hollander, Douglas A. Wolfe, and Eric Chicken. 2013. *Nonparametric Statistical Methods* (3rd ed.). John Wiley & Sons.
- [17] Di Jin, Bunyamin Sisman, Hao Wei, Xin Luna Dong, and Danai Koutra. 2021. Deep Transfer Learning for Multi-source Entity Linkage via Domain Adaptation. *Proc. VLDB Endow.* 15, 3 (2021), 465–477. doi:10.14778/3494124.3494131
- [18] Nishadi Kirielle, Peter Christen, and Thilina Ranbaduge. 2022. TransER: Homogeneous Transfer Learning for Entity Resolution. In *Proceedings of the 25th International Conference on Extending Database Technology, EDBT, OpenProceedings.org*, 2:118–2:130. doi:10.48786/EDBT.2022.03
- [19] Pradap Konda, Sanjib Das, Paul Suganthan G. C., AnHai Doan, Adel Ardalan, Jeffrey R. Ballard, Han Li, Fatemah Panahi, Haojun Zhang, Jeffrey F. Naughton, Shishir Prasad, Ganesh Krishnan, Rohit Deep, and Vijay Raghavendra. 2016. Magellan: Toward Building Entity Matching Management Systems. *Proc. VLDB Endow.* 9, 12 (2016), 1197–1208. doi:10.14778/2994509.2994535
- [20] Hannah Köpcke, Andreas Thor, and Erhard Rahm. 2010. Learning-Based Approaches for Matching Web Data Entities. *IEEE Internet Computing* 14, 4 (2010), 23–31. doi:10.1109/MIC.2010.58
- [21] Stefan Lerm, Alieh Saeedi, and Erhard Rahm. 2021. Extended Affinity Propagation Clustering for Multi-source Entity Resolution. In *Datenbanksysteme für Business, Technologie und Web (BTW (LNI, Vol. P-311)*. Gesellschaft für Informatik, Bonn, 217–236. doi:10.18420/BTW2021-11
- [22] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2020. Deep Entity Matching with Pre-Trained Language Models. *Proc. VLDB Endow.* 14, 1 (2020), 50–60. doi:10.14778/3421424.3421431
- [23] Robin Linacre, Sam Lindsay, Theodore Manassis, Zoe Slade, Tom Hepworth, Ross Kennedy, and Andrew Bond. 2022. Splink: Free software for probabilistic record linkage at scale. *International Journal of Population Data Science* 7, 3 (Aug. 2022). doi:10.23889/ijpds.v7i3.1794
- [24] David Lopez-Paz and Maxime Oquab. 2016. Revisiting classifier two-sample tests. *CoRR abs/1610.06545* (2016). arXiv:1610.06545 <http://arxiv.org/abs/1610.06545>
- [25] Michael Loster, Ioannis K. Koumarelas, and Felix Naumann. 2021. Knowledge Transfer for Entity Resolution with Siamese Neural Networks. *ACM J. Data Inf. Qual.* 13, 1 (2021), 2:1–2:25. doi:10.1145/3410157
- [26] Jakub Maciejewski, Konstantinos Nikolettos, George Papadakis, and Yannis Velegarakis. 2025. Progressive Entity Matching: A Design Space Exploration. *Proceedings ACM Management Data* 3, 1 (2025). doi:10.1145/3709715
- [27] Barzan Mozafari, Purna Sarkar, Michael Franklin, Michael Jordan, and Samuel Madden. 2014. Scaling Up Crowd-sourcing to Very Large Datasets: A Case for Active Learning. *PVLDB Endowment* 8, 2 (Oct. 2014), 125–136.
- [28] 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*. ACM, 19–34. doi:10.1145/3183713.3196926
- [29] Axel-Cyrille Ngonga Ngomo and Klaus Lyko. 2012. EAGLE: Efficient Active Learning of Link Specifications Using Genetic Programming. In *The Semantic Web: Research and Applications*. Berlin, Heidelberg, 149–163.
- [30] Shivani Padmanabhan, Lucy Carty, Ellen Cameron, Rebecca E. Ghosh, Rachael Williams, and Helen Strongman. 2019. Approach to record linkage of primary care data from Clinical Practice Research Datalink to other health-related patient data: overview and implications. *European Journal of Epidemiology* 34, 1 (Jan 2019), 91–99. doi:10.1007/s10654-018-0442-4 Epub 2018 Sep 15.
- [31] George Papadakis, Dimitrios Skoutas, Emmanouil Thanos, and Themis Palpanas. 2021. Blocking and Filtering Techniques for Entity Resolution: A Survey. *ACM Comput. Surv.* 53, 2 (2021), 31:1–31:42. doi:10.1145/3377455
- [32] Ralph Peeters and Christian Bizer. 2023. Using ChatGPT for Entity Matching. In *New Trends in Database and Information Systems - ADBIS 2023 (Communications in Computer and Information Science, Vol. 1850)*. Springer, 221–230. doi:10.1007/978-3-031-42941-5\_20
- [33] Anna Primpeli and Christian Bizer. 2021. Graph-Boosted Active Learning for Multi-source Entity Resolution. In *The Semantic Web - ISWC 2021 - 20th International Semantic Web Conference, ISWC (Lecture Notes in Computer Science, Vol. 12922)*. Springer, 182–199. doi:10.1007/978-3-030-88361-4\_11
- [34] Anna Primpeli, Ralph Peeters, and Christian Bizer. 2019. The WDC Training Dataset and Gold Standard for Large-Scale Product Matching. In *Companion of The 2019 World Wide Web Conference, WWW*. ACM, 381–386. doi:10.1145/3308560.3316609
- [35] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. 2007. Near linear time algorithm to detect community structures in large-scale networks. *Phys. Rev. E* 76 (Sep 2007), 036106. Issue 3. doi:10.1103/PhysRevE.76.036106
- [36] Alieh Saeedi, Lucie David, and Erhard Rahm. 2021. Matching Entities from Multiple Sources with Hierarchical Agglomerative Clustering. In *IC3K*. SCITEPRESS, 40–50. doi:10.5220/0010649600003064

- [37] Alieh Saeedi, Eric Peukert, and Erhard Rahm. 2018. Using Link Features for Entity Clustering in Knowledge Graphs. In *The Semantic Web - 15th International Conference, ESWC 2018, Proceedings (Lecture Notes in Computer Science, Vol. 10843)*. Springer, 576–592. doi:10.1007/978-3-319-93417-4\_37
- [38] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. CoRR abs/1910.01108 (2019). arXiv:1910.01108 <http://arxiv.org/abs/1910.01108>
- [39] Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.* 45, 11 (1997), 2673–2681. doi:10.1109/78.650093
- [40] Naeem Siddiqi. 2005. *Credit Risk Scorecards: Developing and Implementing Intelligent Credit Scoring*. John Wiley & Sons.
- [41] Michael Stonebraker and Ihab F. Ilyas. 2018. Data Integration: The Current Status and the Way Forward. *IEEE Data Eng. Bull.* 41, 2 (2018), 3–9.
- [42] 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. *Proc. VLDB Endow.* 14, 11 (2021), 2459–2472. doi:10.14778/3476249.3476294
- [43] Vincent A. Traag, Ludo Waltman, and Nees Jan van Eck. 2019. From Louvain to Leiden: guaranteeing well-connected communities. *Scientific Reports* 9, 1 (March 2019). doi:10.1038/s41598-019-41695-z
- [44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*. 5998–6008. <https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html>
- [45] Cédric Villani. 2009. *Optimal Transport: Old and New*. Springer-Verlag Berlin Heidelberg. doi:10.1007/978-3-540-71050-9
- [46] Qing Wang, Dinusha Vatsalan, and Peter Christen. 2015. Efficient Interactive Training Selection for Large-Scale Entity Resolution. In *PAKDD*. Ho Chi Minh City, Vietnam.
- [47] Runhui Wang, Yuliang Li, and Jin Wang. 2023. Sudowoodo: Contrastive Self-supervised Learning for Multi-purpose Data Integration and Preparation. In *39th IEEE International Conference on Data Engineering, ICDE*. IEEE, 1502–1515. doi:10.1109/ICDE55515.2023.00391
- [48] Timothé Watteau, Aubin Bonnefoy, Simon Illouz-Laurent, Joaquim Jusseau, and Serge Iovleff. 2024. Advanced Graph Clustering Methods: A Comprehensive and In-Depth Analysis. arXiv:2407.09055 <https://arxiv.org/abs/2407.09055>
- [49] Renzhi Wu, Sanya Chaba, Saurabh Sawlani, Xu Chu, and Saravanan Thirumuruganathan. 2020. ZeroER: Entity Resolution using Zero Labeled Examples. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference*. ACM, 1149–1164. doi:10.1145/3318464.3389743
- [50] Xiaocan Zeng, Pengfei Wang, Yuren Mao, Lu Chen, Xiaozhe Liu, and Yunjun Gao. 2024. MultiEM: Efficient and Effective Unsupervised Multi-Table Entity Matching. In *40th IEEE International Conference on Data Engineering, ICDE*. IEEE, 3421–3434. doi:10.1109/ICDE60146.2024.00264
- [51] Zeyu Zhang, Paul Groth, Iacer Calixto, and Sebastian Schelter. 2025. A Deep Dive Into Cross-Dataset Entity Matching with Large and Small Language Models. In *Proceedings 28th International Conference on Extending Database Technology, EDBT*. OpenProceedings.org, 922–934. doi:10.48786/EDBT.2025.75
- [52] 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, WWW*. ACM, 2413–2424. doi:10.1145/3308558.3313578
- [53] Xiaojin Zhu and Zoubin Ghahramani. 2002. *Learning from labeled and unlabeled data with label propagation*. Technical Report CMU-CALD-02-107. Carnegie Mellon University.