IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

*Digital Object Identifier 10.1109/ACCESS.2017.DOI*

# TransClean: Finding False Positives in Multi-Source Entity Matching under Real-World Conditions via Transitive Consistency

**FERNANDO DE MEER PARDO[1,2], BRANKA HADJI MISHEVA[3], MARTIN BRASCHLER[1] AND KURT STOCKINGER[1]**

[1]ZHAW Zurich University of Applied Sciences, Winterthur, Switzerland
[2]University of Zurich, Zurich, Switzerland
[3]Bern University of Applied Sciences, Bern, Switzerland

## ABSTRACT

We present TransClean, a method for detecting false positive predictions of entity matching algorithms under real-world conditions characterized by large-scale, noisy, and unlabeled multi-source datasets that undergo distributional shifts. TransClean is explicitly designed to operate with multiple data sources in an efficient, robust and fast manner while accounting for edge cases and requiring limited manual labeling. TransClean leverages the *Transitive Consistency*, a measure that we propose, aimed to detect the lack of consistency of a pairwise matching model $f_\theta$ on the graph $G_{f_\theta}$ implied by its predicted pairwise matches. The *Transitive Consistency* is calculated via all the predictions of $f_\theta$ on the pairs of records belonging to the same connected components in $G_{f_\theta}$. TransClean iteratively modifies a matching through gradually removing false positive matches while removing as few true positive matches as possible. In each of these steps, the estimation of the *Transitive Consistency* is exclusively done through model inference and produces indicators that can be used as proxies of the amounts of true and false positives in the matching while not requiring any manual labeling. These indicators produce estimates of the quality of the matching and point out which record groups are likely to contain false positives. In our experiments, we compare combining TransClean with a naively trained pairwise matching model (DistilBERT) and with a state-of-the-art end-to-end matching method (CLER) and illustrate the flexibility of TransClean in being able to detect most of the false positives of either setup across a variety of datasets. Our experiments show that TransClean induces an average +24.42 F1 score improvement for entity matching in a multi-source setting when compared to traditional pair-wise matching algorithms.

## I. INTRODUCTION

*Entity Matching* (EM) describes the process in which data records originating from different sources are compared to find those that refer to the same *real-world entities*. EM is a vital part of any *Extract-Transform-Load (ETL)/data integration* process, as it enables the joint use of all available data items relating to each real-world entity when such items originate from different sources. Successful EM systems allow for the merging of different data sources into unified repositories and prevent inaccuracies caused by duplicate records. EM, itself ironically referred to via multiple names such as *Record Linkage* (RL), *Entity Resolution* (ER) and *Data Integration/Deduplication*, has been studied since the 1940s [13], [15] and has been classically tackled via rule-based algorithms and heuristics [9], [20], [25], [36]. In recent years, however, most state-of-the-art EM methods rely on machine learning (ML), most notably Transformer models [6], [29], [30], [53].

ML-based methods frame EM as a *binary classification* problem by classifying pairs of records as either `Match` or `NoMatch` and are usually compared to each other based exclusively on their performance on this task. However, based

on individual pairwise predictions, entity matching models also implicitly define groupings of records through transitive (implicit) links. These transitive relationships play a crucial role in multi-source settings, yet they have been overlooked by previous methods since they are not reflected in scores based only on pairwise binary predictions. We will discuss them in detail in Section IV.

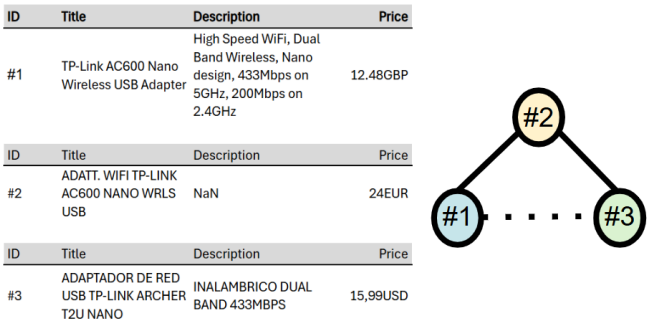


**FIGURE 1.** Example of a matching involving three records from three different data sources. The matches ($r_{\#1}$, $r_{\#2}$) and ($r_{\#2}$, $r_{\#3}$) imply also the transitive match ($r_{\#1}$, $r_{\#3}$) even if $r_{\#1}$ and $r_{\#3}$ have not been directly paired.

Consider, for instance, the matching of records $r_{\#1}$, $r_{\#2}$ and $r_{\#3}$ from three different data sources as shown in Figure 1. If an EM method predicts pairs ($r_{\#1}$,$r_{\#2}$) and ($r_{\#2}$,$r_{\#3}$) as matches, then it is implying the match ($r_{\#1}$,$r_{\#3}$) as well (see dotting line in Figure 1). Such matches have been referred to as *transitive matches* [12] and can be present in any matching setting - empheven in the standard setting of matching records from only two different sources. Transitive matches are, however, most problematic in the multi-source matching case, due to the greater numbers of candidate pairs being considered and the propagating effect of false positive predictions when wrongly linking different record groups. This is because two wrongly linked record groups lead to large numbers of false positive transitive matches since each implied transitive match between records of different groups will be a false positive.

In the real-world, EM tasks involve numerous large unlabeled datasets which make evaluating the performance of different EM methods and comparisons among them unfeasible in practice, as verifying the faithfulness of a given set of pairwise matches requires manually labeling them one by one, which is extremely time consuming when dealing with millions of record pairs. Effectively, the evaluation of a given matching across unlabeled data sources is in itself as challenging as producing the matching itself, if not even more so. Comparisons between different matchings can be attempted via analyzing, for example, statistics of the produced record groups such as the size distribution, the number of pairwise predictions per group, etc. Such comparisons, however, are not conclusive and cannot alone pinpoint the strengths and weaknesses of each EM method unless extensive manual labeling is also involved.

This difficulty has created a significant gap between EM research and real-world projects. On one hand, EM research is focused on improving state-of-the-art black box pairwise prediction models where ground truth matches are available. On the other hand, in real-world projects, EM tasks are usually tackled through heuristics on a case-by-case basis [2], [52], where ground truth matches are typically not available or very expensive to get. The narrowing of this gap can only be achieved through developing EM methods that account for several criteria such as:

- **Manual labeling:** Only a reduced number of candidate record pairs can be expected to be manually labeled by human annotators. As a task, manual labeling is inherently labor-intensive and monotonous, often leading to annotator fatigue, which negatively impacts labeling accuracy over time [35].
- **Run time:** EM methods need to produce matchings in a reasonable time, the complexity of every step of an EM pipeline needs to be carefully accounted for.
- **Correctness assurance:** Black box EM methods should provide some form of additional evidence on the correctness of the matching produced beyond the predictions themselves, so that extensive manual labeling can be avoided.

Our aim is to add to this line of research by presenting the concept of *Transitive Consistency*, a measure of the consistency of a pairwise matching model $f_\theta$ based on its predictions on the *transitive matches* of its produced matching $G_{f_\theta}$. We also introduce *TransClean*, a technique that leverages the Transitive Consistency to identify and remove false positive predictions through a series of iterative steps and accounts for all the criteria listed previously. Our main contributions are as follows:

1) We introduce the concept of Transitive Consistency and TransClean, a method that can be combined with any pairwise matching model and aims to detect the false positive predictions of a given matching. TransClean is inspired by a large real-world use case where entities of multiple sources need to be matched. Hence, TransClean is explicitly designed for the matching of multiple, large and unlabeled datasets in an efficient, robust and fast manner while accounting for edge cases and requiring little manual labeling.
2) In our experiments, we combine TransClean both with a naively trained pairwise matching model (DistilBERT) and a state-of-the-art EM method (CLER [53]) across a variety of datasets and illustrate TransClean's ability to detect large proportions of the false positives produced by both models.
3) We illustrate the advantages of using TransClean over using exclusively traditional pairwise matching algorithms. Specifically, we show how the Transitive Consistency, calculated via the predictions on transitive matches, can be used as an observable proxy of the quality of a matching and how its evolution during

the different steps of TransClean indicates a gradual improvement of the matching, eliminating the need to perform extensive manual labeling. Our experiments show that TransClean induces an average +24.42 F1 score improvement for entity matching in a multi-source setting when compared to traditional pair-wise matching algorithms.

The manuscript is organized as follows. Section II discusses the related work on EM with ML methods. Section III presents a discussion on the challenges that arise in real-world EM tasks and the issues associated with using exclusively pairwise matching models. Section IV presents the definition of Transitive Consistency and the different steps that make up TransClean. Section V discusses the setup and results of the different entity matching experiments we run across multiple datasets. Finally, Section VI presents our conclusions along with an outlook on future work.

## II. RELATED WORK

Current state-of-the-art EM methods employ ML models based on variations of the Transformer architecture [48]. Classical EM methods based on combinations of hand-crafted heuristics [25], [36] have been overtaken by ML-based ones due to the ability of the former at overcoming common EM challenges such as noisy and heterogeneous records (e.g. with missing data, misplaced attributes etc.) by treating them as instances of textual data. Through fine-tuning, pre-trained Transformer models are able to learn complex mappings from sets of manually labeled pairs.

Following this line of thought, multiple works present different training optimizations, such as pseudo-labeling, data augmentation and contrastive learning, that aim to improve the performance of pairwise pre-trained Transformer classifiers [6], [18], [29], [32], [37], [40], [49], [54], often paired with Active Learning techniques [2], [5], [14], [16], [17], [19], [24], [28], [47], [57] and Large Language Models (LLMs) [33], [39], [50], [56][1]. More recently, even quantum neural networks have been used for entity matching [4].

These methods are usually compared based on their performance on multiple benchmarks consisting of 2 data sources such as those published by the Database Group of the University of Leipzig [26] and the Magellan datasets [11] which have fixed train/val/test splits of record pairs. This evaluation methodology contributes to the research/real-world gap since in the former, no fixed sets of candidate pairs are available for matching, instead one has to select which subset of pairs to be evaluated by the pairwise matching model, a step commonly referred to as *blocking*.

Blocking is a key part of EM as the set of all possible pairs grows quadratically w.r.t. the number of records being matched, which makes the evaluation of all possible pairs unfeasible for large datasets. This is, however, ignored when using fixed splits of record pairs to evaluate the performance of different EM methods. There are works that focus exclusively on blocking, also framing it as a ML problem [46], [55]. Other works such as Sudowoodo [49] and CLER [53] perform both blocking and matching while also considering labeling budgets, which make them more applicable to real-world scenarios.

All of the previous methods consider only the pairwise model predictions in their evaluations but, as we previously discussed, matching involves producing record groups which contain both the explicit and the implicit/transitive matches produced by a model. Previous works which also consider records groups and carry out matching in an agglomerative fashion include techniques such as *Clustering* [20], [34], [41]–[43] and *Progressive ER/EM* [1], [31], [45], [51]. Other works focus on model mistakes and propose methods to rank the most important model predictions for manual labeling [7], [8], [21]. Such rankings, however, are of limited practical use in the matching of multiple large unlabeled datasets, since the amount of selected pairs to label may still be overwhelming to inspect. In addition, earlier approaches such as collective matching using relational constraints ( [3], [22], [27]), aimed to enforce consistency across record groups using joint inference. While less scalable, these methods inspired consistency-based ideas such as our Transitive Consistency framework.

We add to these previous lines of work by presenting TransClean, a technique that explicitly aims to discover false positive predictions present in a matching with a limited labeling effort and provides information of the quality of a matching without any human intervention.

## III. REAL-WORLD MATCHING CONDITIONS

In most real-world scenarios, a standard way to attempt to match several data sources with a pairwise matching model would be as follows:

1) Initially, a portion of matches would be selected to be manually labeled.
2) One may divide this set of labeled pairs into `train/val/test` splits, fine-tune with the `train` split and select a set of model weights based on the performance on the `val` split.
3) Finally, evaluate the pairs of the `test` split to estimate the performance of the model on unseen pairs.

However, when using the selected model to match the entire dataset (via evaluating the candidate pairs resulting from applying a blocking to all unlabeled records), the size of the matching produced can be expected to be of *magnitudes higher* than that of the labeled subset of pairs previously used for fine-tuning. This effectively makes evaluating the matching unfeasible, as doing so requires manually checking all of its matches. This is especially problematic in real-world scenarios where significant distributional shifts can lead to large numbers of erroneously predicted matches, either due to the difficulty inherent to the records themselves, or to data artifacts that lead to edge cases.

[1]LLM-based works however present concerns regarding test data leakage during pre-training (as EM benchmark datasets are public and thus likely part of pre-training corpora) which question reported scores and generalization capabilities

Some characteristics of the matching produced, such as the number of records per group, can be used to detect which record groups are likely to have false positive predictions, which tend to be the biggest produced groups due to the agglutinative effect of false positive matches [12]. This would, however, be quite a rudimentary methodology to detect false positives, since it requires manually inspecting all the matches of record groups in descending order, saving little in terms of manual labeling effort, and provides no information about the quality of smaller record groups which are likely the most numerous. **The challenge is then to develop a technique capable of effectively identifying a majority of the** *false positive predictions* **of a given matching, which preserves the largest amount of** *true positive predictions***. Moreover, the technique should operate with a** *limited manual labeling budget* **and provide some sort of assurance on the correctness of the unchecked record groups.**

## IV. METHODOLOGY

Entity Matching requires considering not only the *explicitly* predicted matches produced by a matching method, but also the *implied* matches i.e. all the non-adjacent pairs of records belonging to the same *component*[2] of the matching graph $G_{f_\theta} = (V, E_{f_\theta})$ whose nodes and edges represent records and predicted matches respectively[3]. This is because all records belonging to the same *component* of $G_{f_\theta}$ are implied to represent the same real-world entity. The actual output of a matching is a series of *cliques*[4] even if not all pairs of vertices of $G_{f_\theta}$ belonging to the same component are adjacent. That is to say, a set of pairwise predictions implicitly leads to a *label propagation* step in which all transitively connected records are assigned to the same group/entity.

Our focus is the detection of *false positives* of pairwise classifiers applied to the problem of multi-source Entity Matching. We aim to do this by analyzing the *consistency* of each classifier/matching model $f_\theta$ with regards to its implicit/transitive matches which emanate from its own explicitly predicted matches (i.e. from $G_{f_\theta}$).

In this section we illustrate how this consistency property can be leveraged by a graph cleanup method that efficiently detects false positives. Additionally, we show how the calculations involved in estimating this property produce quantities that indicate the quality of a given matching without having access to the ground truth.

### A. TRANSITIVE CONSISTENCY

**Definition. Transitive Match:** Let $G_{f_\theta} = (V, E_{f_\theta})$ be a matching graph where $V$ is the set of records to be matched and $E_{f_\theta}$ are the predicted matches $(r_i, r_j)$ of a pairwise matching model $f_\theta$. Two nodes $r_{i*}$ and $r_{j*}$ are said to form an *implicit* or *transitive match* $(r_{i*}, r_{j*})$ of $G_{f_\theta}$ if they are non-adjacent while belonging to the same component of $G_{f_\theta}$.

Let us reconsider the example matching graph shown in Figure 1. In this case, the nodes $r_{\#1}$ and $r_{\#3}$ are a transitive match since they are non-adjacent, i.e. there is no direct edge between them.

**Definition. Transitive Consistency:** Let $G_{f_\theta} = (V, E_{f_\theta})$ be a matching graph where $V$ is the set of records to be matched and $E$ is the set of predicted matches $(r_i, r_j)$ by a pairwise matching model $f_\theta$. We say that $f_\theta$ is *transitively consistent* in $G_{f_\theta}$ if for every transitive match $(r_{i*}, r_{j*})$ of $G_{f_\theta}$:

$$f_\theta((r_{i*}, r_{j*})) = \texttt{Match}$$

In the example of records $r_{\#1}$, $r_{\#2}$ and $r_{\#3}$ shown in Figure 1, a given pairwise matching model $f_\theta$ is transitively consistent if given that $f_\theta((r_{\#1}, r_{\#2})) = \texttt{Match}$ and $f_\theta((r_{\#2}, r_{\#3})) = \texttt{Match}$ then $f_\theta((r_{\#1}, r_{\#3})) = \texttt{Match}$ as well. Intuitively, the Transitive Consistency is a check of whether a pairwise matching model $f_\theta$ fully agrees on all the possible pairs in each component it has produced and if it does not, it points out in which record group/component the discrepancy arises.

Figure 2 illustrates an example record group implied by a set of pairwise predictions with some of its transitive matches. The records are from five different sources (indicated by different colors of the nodes). The transitive match ($r_{\#1}$, $r_{\#4}$) is a true match because it involves records transitively connected by a path made up exclusively of true positive predictions (shown as black edges). In turn, ($r_{\#5}$, $r_{\#10}$) and ($r_{\#6}$, $r_{\#9}$) are non-matches because all the paths between said records include false positive predictions (shown as orange edges). In order for $f_\theta$ to be *transitively consistent* it would have to predict all possible pairs between records of the component as a `Match`.

Note that not all transitive matches $(r_{i*}, r_{j*})$ of each component need to be evaluated in order to check for the Transitive Consistency of a model since a single `NoMatch` prediction suffices. Nonetheless, evaluating all transitive matches provides an estimation on the number of inconsistencies per record group and overall in the matching. We will refer to the transitive matches predicted as `Match` as *positive transitive predictions* and those predicted as `NoMatch` as *negative transitive predictions*.

Contrary to the true and false positives of a matching $G_{f_\theta} = (V, E_{f_\theta})$, the calculation of the Transitive Consistency of a model $f_\theta$ depends only on model predictions and does not require any manual labeling since, by definition, it is independent of the ground truth. Negative transitive predictions are very likely in components/record groups with false positive matches, both because false positive predictions link unrelated records and because they tend to create larger record groups than true positives and thus produce more transitive matches to be evaluated. Because of this, positive

[2]A component of an undirected graph is a connected subgraph that is not part of any larger connected subgraph. Here, and throughout the whole manuscript, we use the term *components* to refer to the produced record groups of the matching given by $G_{f_\theta}$.

[3]Throughout the manuscript we will refer to records as nodes and edges as matches, pairs or predictions interchangeably.

[4]A clique is a subset of vertices of an undirected graph such that every two distinct vertices in the clique are adjacent. Throughout the manuscript we refer to cliques as record groups and vice versa.

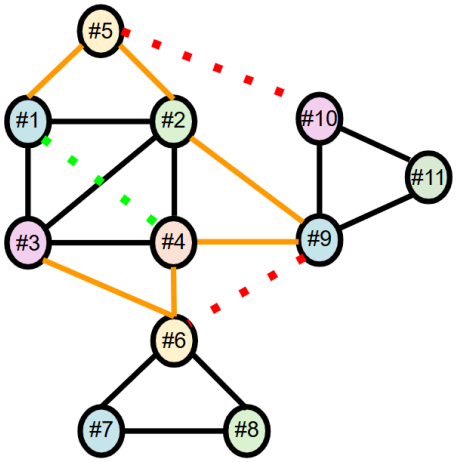


**FIGURE 2.** Example record group implied by a set of pairwise predictions between records $\{r_i\}_{i=1}^{11}$ originating from 5 different data sources. Black edges illustrate true positive predictions, orange edges false positive predictions, red dotted edges denote two of the *transitive matches* (not all possible transitive matches are drawn) between wrongly linked records (due to the false positive predictions) and the green dotted edge denotes a *transitive match* between correctly linked records.

and negative transitive predictions can be used as a proxy of true and false positives of a matching, as we will show in the experiments of Section V.

The Transitive Consistency becomes more informative as more data sources are introduced, since each additional source contributes candidate pairs that, when predicted as matches, generate new transitive matches. The more transitive matches in a given record group, the more informative the Transitive Consistency of said group becomes, as it requires the agreement of the pairwise matching model on a bigger number of record pairs. This turns the matching of additional data sources, which has classically been a challenge due to the increased likelihood of erroneous pairwise predictions, into an advantage.

### B. TRANSCLEAN

In this section we present TransClean, a cleanup technique that leverages the Transitive Consistency of a pairwise matching model $f_\theta$ to identify false positives in a given matching graph $G_{f_\theta} = (V, E_{f_\theta})$.

#### 1) Introductory Description

TransClean consists of three different steps which can be described as follows:

- **First Step** (*Initial Step with Finetuning*)**:** In the initial step the goal is to detect the most easily identifiable false positive pairwise predictions. We do this through selecting and labeling records from the components with the biggest numbers of false positive transitive predictions since they indicate the most flagrant violations of the Transitive Consistency. Additionally, we use the labeled pairs to further finetune $f_\theta$.
- **Second Step** (*Post Finetuning Cleanup & Checks*)**:** The second step aims to remove from the matching as many false positives as possible. We do this by removing records from all components that violate the Transitive Consistency through a series of checks (which involve labeling) and a removal heuristic (which does not label removed pairs).
- **Final Step** (*Edge Recovery*)**:** The final step considers adding back some of the removed pairs (edges) while maintaining the Transitive Consistency of the resulting matching. This is done by evaluating the transitive matches that would be created if removed edges were to be added back to the matching.

TransClean is designed to operate within a predefined labeling budget $LB^{Total}$. The labeling can be done either manually or using pseudo-labels. In the experiments we will illustrate the performance difference between using an LLM to label pairs and manually labeling them.

We now give detailed descriptions with pseudo-code of each of the steps of TransClean. See Figure 3 visual flowchart of the entire cleanup procedure and Figure 6 for an illustration of how TransClean would process the component of Figure 2.

#### 2) Initial Steps with Finetuning

In the initial stages of TransClean, our primary goal is to detect the false positives that have the biggest effect in terms of wrongly linking different record groups in $G_{f_\theta}$ (such as the pairs $(r_{\#2}, r_{\#9})$/$(r_{\#4}, r_{\#9})$ and $(r_{\#3}, r_{\#6})$/$(r_{\#4}, r_{\#6})$ of Figure 2). Detecting such matches and labeling them will also allow us to further fine-tune the pairwise matching model, potentially making it able to detect similar mistakes in other record groups.

To start off, we evaluate with $f_\theta$ all the transitive matches of $G_{f_\theta}$[5] and sort its components by their number of negative transitive predictions. Next, we select from each component:

- **Edges belonging to a** *Minimum Edge Cut*. Edges belonging to a *Minimum Edge Cut* make up a set of minimum cardinality that if removed would disconnect the component. Minimum Edge Cuts of components with large numbers of negatively predicted transitive edges are likely to contain false positives. Calculating the Minimum Edge cut of large components can be costly since its complexity is $O(V_c^2 E_c)$ for a component $c = (V_c, E_c)$ [23]. However, keep in mind that we only calculate minimum edge cuts for specific components so naturally $|V_c| << |V|$, $|E_c| << |E_{f_\theta}|$ where the notation $|\cdot|$ denotes the size of a set and $(V, E_{f_\theta})$ are the sets' nodes and edges of the entire matching graph $G_{f_\theta}$. Nonetheless, to prevent the calculation of the Minimum

[5]If a component has more nodes than the threshold $S$, we only evaluate a subset of its transitive matches to avoid long running times, as the number of transitive matches grows quadratically with the number of nodes of the component.

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2025.3632400



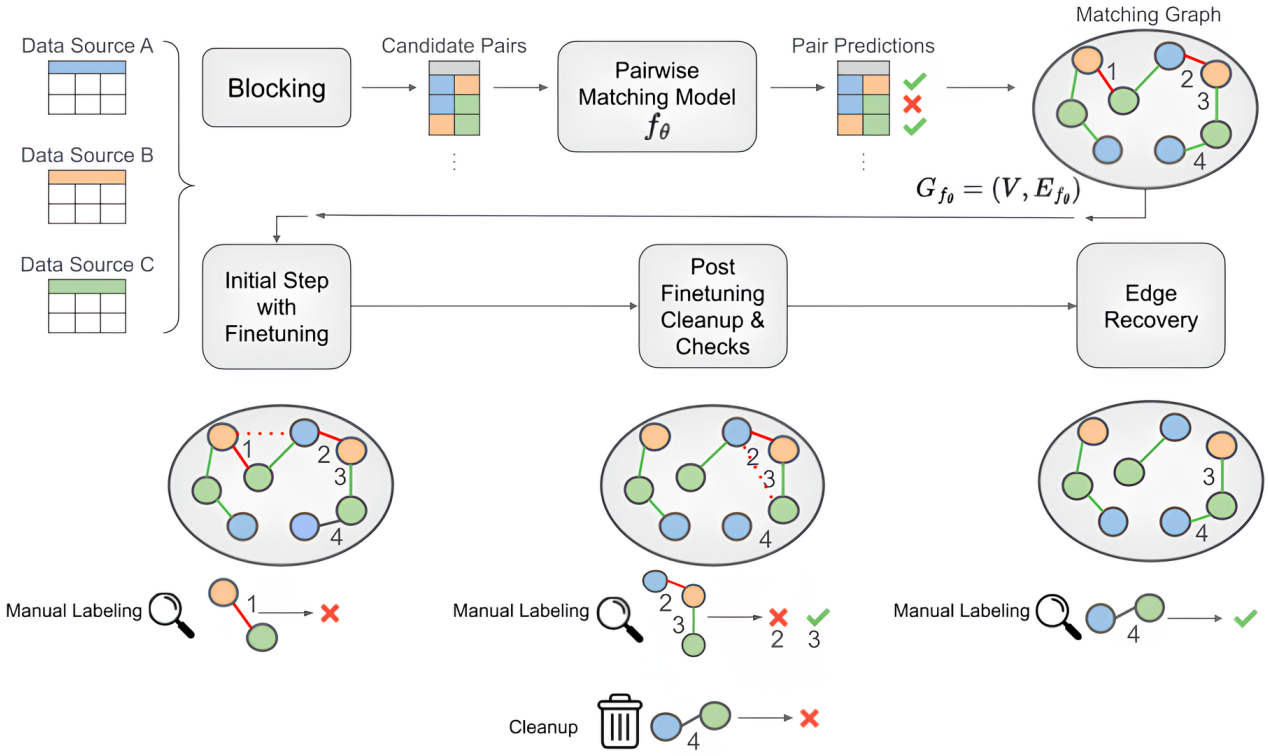


**FIGURE 3.** Visual flowchart illustrating the application of TransClean to a matching graph $G_{f_\theta} = (V, E_{f_\theta})$ with true positive edges (colored in green) and false positive edges (colored in red). In the *Initial Step with Finetuning* a negative transitive prediction (dotted red line) leads to the manual labeling of false positive edge 1. In the *Post Finetuning Cleanup & Checks* edges 2 and 3 are manually checked (because of another negative transitive prediction, again depicted as a dotted red line), leading to the removal of false positive edge 2, while true positive edge 4 is removed due to a cleanup step. Finally, the *Edge Recovery* step recovers removed true positive edge 4. The flowchart illustrates TransClean's objective: Finding and removing false positive edges of a matching graph, under a limited labeling budget and removing as few true positives as possible.

Edge Cuts of exceedingly large components, we initially remove from the matching all edges of components bigger than a threshold $S$.

- **Edges constituting** *shortest paths* **between the records of negatively predicted transitive matches in the component**. If a transitive edge is predicted negatively then it is likely to be a true negative pair. Since there cannot be an edge path of only true positives connecting the two records of a true negative pair, selecting edges in this manner will likely yield false positives.

We carry out this edge selection and labeling process, removing from $G_{f_\theta}$ the edges labeled as false positives, until we reach a given labeling budget. Once the budget has been reached, we fine-tune $f_\theta$ with the edges that have been labeled (either manually or with pseudo-labels).

Additionally, we want to start leveraging the Transitive Consistency of $f_\theta$ in order to detect the inconsistent components of $G_{f_\theta}$ and break them up. To do this, we evaluate all the current transitive matches of $G_{f_\theta}$ with the fine-tuned model $f_{\theta^{TC}}$ and remove all the edges belonging to the Minimum Edge Cuts of components with more negative than positive transitive predictions. This pruning logic naturally decreases the number of negative transitive predictions, since the removal of Minimum Edge Cuts disconnects the sets of nodes which previously were creating the transitive matches. We expect to remove significantly more false positives than true positives with this logic, but the removal of some true positives is the price to pay for not labeling all the pairs considered this way. We repeat all of these steps $n$ times, accumulating the newly labeled pairs at every step.

See Algorithm 1 for the pseudo-code description of this 1st step and Figure 4 for an illustration of the edge selection process. All successive steps carry out the same steps but starting out with the fine-tuned model $f_{\theta^{TC}}$, the modified matching graph $G^{TC}$, and the labeled pairs $FT_{pairs}$ instead.

### 3) Post Finetuning Cleanup & Checks

After completing the initial TransClean stages, we expect to have removed most of the easily detectable false positives of the matching. At this point we have a model $f_{\theta^{TC}}$ that has been repeatedly fine-tuned and a modified matching graph $G^{TC}$ which has had the labeled false positive edges removed. Even though we have repeatedly removed Minimum Edge Cuts from the components with more negative transitive predictions than positive ones, there might still be components that fulfill this criterion, i.e. components with more

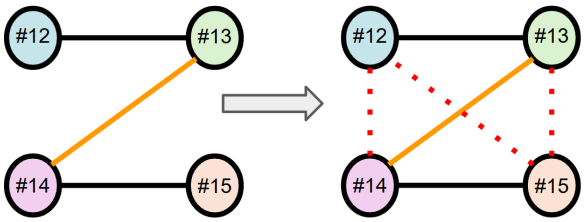


**FIGURE 4.** The component in this figure has two true positive matches (black edges) and a single false positive match between records $r_{\#13}$ and $r_{\#14}$ (orange edge). If any of the transitive edges $(r_{\#12}, r_{\#14})$, $(r_{\#12}, r_{\#15})$ and $(r_{\#13}, r_{\#15})$ (red dotting edges) is predicted as a `NoMatch`, then the model will not be transitively consistent in the component. In this case, all edges may be selected during Algorithm 1. Either for fine-tuning, as a Minimum Edge Cut or part of a shortest path, or to be pruned without being labeled, also as a Minimum Edge Cut.

**Algorithm 1** 1st Initial TransClean Step with Finetuning

**Input:** Pairwise matching model $f_\theta$, matching graph $G_{f_\theta} = (V, E_{f_\theta})$, labeling budget $LB^{init}$, component size threshold $S$

**Output:** Finetuned model $f_{\theta^{TC}}$, modified $G^{TC}$

$C = \{c_1, c_2, ..., c_n\}$ components of $G_{f_\theta}$ sorted by size
$C \leftarrow LargeSubgraphBreakdown(C, S)$
$G^{TC} \leftarrow C$ *#Initialize the matching graph*
$f_{\theta^{TC}} \leftarrow f_\theta$ *#Initialize the matching model*
$LB_{spent} \leftarrow 0$ *#Initialize the labeling budget*
$FT_{pairs} \leftarrow \emptyset$ *#Initialize the set of finetuning pairs*
$i \leftarrow 0$
**while** $LB_{spent} < LB^{init}$ **do** and $i < n$:
    $E_{\text{mincut}} \leftarrow$ MinEdgeCut($c_i$) *#Select the Minimum Edge Cut pairs*
    $E_{ShortestPaths} \leftarrow$ ShortestPaths($c_i$) *#Select the shortest paths pairs*
    $FT_{pairs} \leftarrow FT_{pairs} \bigcup E_{\text{mincut}} \bigcup E_{ShortestPaths}$ *#Accumulate the finetuning pairs*
    $G^{TC} \leftarrow Label(G^{TC}, FT_{pairs})$
    $LB_{spent} \leftarrow LB_{spent} + |E_{\text{mincut}} \bigcup E_{ShortestPaths}|$
    $i \leftarrow i + 1$
$f_{\theta^{TC}} \leftarrow FineTune(f_{\theta^{TC}}, FT_{pairs})$ *#Finetune $f_{\theta^{TC}}$ with the current finetuning pairs*
$G^{TC} \leftarrow Prune_{MinEdgeCut}(G^{TC}, f_{\theta^{TC}})$ *# Prune edges of the components with negatively predicted transitive matches*
**Return:** $f_{\theta^{TC}}, G^{TC}$

negative transitive predictions than positive ones. To make sure we break up these components, we iteratively remove from $G^{TC}$ the Minimum Edge Cuts of components with more negative transitive predictions than positive ones. We repeat this process until all the remaining components have more positive transitive predictions than negative ones.

At this point, we want to exploit the Transitive Consistency to its fullest in order to find the remaining false positives of $G^{TC}$. First, we check and label all the edges belonging to components with more nodes than the threshold $S$ mentioned in the previous section. Following the size check, we check whether any of the current transitive matches has been at any point labeled as a false positive. If any such match is currently implied by the remaining edges in $G^{TC}$, then there will be at least one false positive among the edges of its correspondent component, so we check and label all edges of components implying such matches and remove any labeled false positive edge.

Finally, in order to impose the Transitive Consistency of $f_{\theta^{TC}}$ in the current matching graph, we evaluate all remaining transitive matches and label all edges of components with any negative transitive predictions. We expect the labeling requirement in this step $LB^{PostFT}$ to be manageable, due to the extensive pruning we have previously performed in the initial steps. The amount of pairs left to check will be dependent on $f_{\theta^{TC}}$ and the current state of $G^{TC}$, unlike the labeling requirement of the initial steps which can be chosen instead. See Algorithm 2 for a pseudo-code description of this stage.

### 4) Edge Recovery

After carrying out all the Post Finetuning Cleanup & Checks we are left with a matching $G^{TC}$ in which $f_{\theta^{TC}}$ is transitively consistent[6]. To achieve this, we have removed edges from our original matching $G_{f_\theta}$ in multiple instances. We expect most of these matches to be correctly removed false positives, but some might be true positives, specially those removed during the initial stages, since the initial focus is to break up large components.

We can once again leverage the Transitive Consistency of $f_{\theta^{TC}}$ in order to recover some of the true positive matches we have removed. We do this by evaluating the additional transitive matches that $G^{TC}$ would present if we added back each of the removed unlabeled edges that $f_{\theta^{TC}}$ currently predicts as a `Match`. If all of the transitive matches that the removed edge would create if added back are predicted as a `Match`, then adding the removed edge would preserve the Transitive Consistency of $f_{\theta^{TC}}$ in $G^{TC}$, so we indeed add it back. If any of the transitive matches are predicted as a `NoMatch` then we manually check the removed pair if the remaining labeling budget $LB^{EdgeRecovery} = LB^{Total} - LB^{init} - LB^{PostFT}$ allows for it, otherwise we ignore it. If the removed edge connects two records with no other matches, it will not create any new transitive match in $G^{TC}$ but adding it back would still respect the Transitive Consistency of $f_{\theta^{TC}}$ in $G^{TC}$, so we choose to label these records as well. Repeating this logic for every removed edge, we can add back to the matching difficult matches. See Algorithm 3 for a pseudo-

[6]With the exception of pairs whose predictions might have changed due to the finetuning of $f_{\theta^{TC}}$ (but whose transitive matches are all predicted as positives) and components that have been fully labeled. We could choose to remove all pairs whose predictions become `NoMatch` at any point during the finetuning process, but this will likely remove numerous true positive pairs.

**Algorithm 2** Post Finetuning Cleanup & Checks

**Input:** Pairwise matching model $f_{\theta TC}$, matching graph $G^{TC}$, component size threshold $S$
**Output:** Modified $G^{TC}$, labeling requirement $LB^{PostFT}$

$C \leftarrow \{c_1, c_2, ..., c_n\}$ components of $G^{TC}$
$Stop_{flag} \leftarrow False$
**while** NOT $Stop_{flag}$ **do**: # *Keep on pruning edges until the positive transitive predictions outnumber the negatives*
    $Stop_{flag} \leftarrow True$
    **for** $c_i \in C$ **do**:
        $Pos_{tr}(c_i) \leftarrow \{(r_{i*}, r_{j*}) \in c_i^{trans}$
                $|f_{\theta TC}((r_{i*}, r_{j*})) = \texttt{Match}\}$
        $Neg_{tr}(c_i) \leftarrow \{(r_{i*}, r_{j*}) \in c_i^{trans}$
                $|f_{\theta TC}((r_{i*}, r_{j*})) = \texttt{NoMatch}\}$
        **if** $|Pos_{tr}(c_i)| > |Neg_{tr}(c_i)|$ **then**
            $G^{TC} \leftarrow Prune_{MinEdgeCut}(c_i)$
            $Stop_{flag} \leftarrow False$
    **Update** $C \leftarrow \{c_1, c_2, ..., c_n\}$ components of $G^{TC}$
$G^{TC}, LB^{SC} \leftarrow SizeCheck(G^{TC}, S)$
$G^{TC}, LB^{LTC} \leftarrow LabeledTransCheck(G^{TC})$ # *Manually check all the components with negatively labeled transitive matches*
$LB^{FinalTransCheck} \leftarrow 0$ # *Initialize the final check labeling effort*
**for** $c_i \in C$ **do**: # *Check all components with remaining negative transitive predictions*
    $Neg_{tr}(c_i) = \{(r_{i*}, r_{j*}) \in c_i^{trans}$
            $|f_{\theta TC}((r_{i*}, r_{j*})) = \texttt{NoMatch}\}$
    **if** $|Neg_{tr}(c_i)| > 0$ **then**
        $G^{TC}, LB^{MC} \leftarrow Label(c_i)$ # *Update the matching graph based on assigned labels and record size of checked component*
        $LB^{FinalTransCheck} \leftarrow LB^{FinalTransCheck} + LB^{MC}$
$LB^{PostFT} \leftarrow LB^{SC} + LB^{LTC} + LB^{FinalTransCheck}$
**Return:** $G^{TC}$, $LB^{PostFT}$

code description of the process and Figure 5 for a visual illustration of an instance of the edge recovery process.

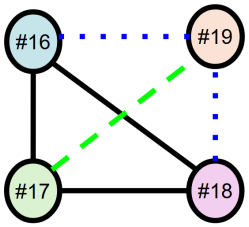


**FIGURE 5.** If we consider to recover the edge between records $r_{\#17}$ and $r_{\#19}$ (predicted as a match by $f_{\theta TC}$), we will evaluate as new transitive matches the record pairs $(r_{\#16}, r_{\#19})$ and $(r_{\#18}, r_{\#19})$. If both are predicted as matches then we add the edge $(r_{\#17}, r_{\#19})$ to the matching, enlarging the component. If instead at least one of the record pairs is predicted as a NoMatch, we will label the edge $(r_{\#17}, r_{\#19})$ if the labeling budget allows for it, otherwise we will ignore it.

Algorithm 3 requires separate evaluations of different sets of transitive matches with the pairwise model $f_{\theta TC}$. All of the potential transitive matches we may need to evaluate can be determined at the beginning of the process, since they are the transitive matches of the components resulting from adding back to the matching **all deleted edges**. For speed considerations all of these evaluations can be carried out in batches, which is the setup most ML hardware is optimized for, but doing so requires carefully considering the size of the considered connected components. In our actual implementation, we use the component size threshold S, used in previous steps, to only evaluate the transitive matches of components smaller than S. We manually check the deleted edges that would lead to bigger components if the labeling budget $LB^{EdgeRecovery}$ allows for it or ignore them otherwise.

Note that adding a true positive edge to $G^{TC}$ may lead to an increase in the number of negative transitive predictions of $f_{\theta TC}$ in $G^{TC}$. This is because the model may wrongly predict true positive transitive matches, as is particularly likely in record groups with edge cases. Negative transitive predictions can be ignored as long as they are caused by manually labeled matches, as the manual inspection constitutes stronger evidence than model predictions. It is only in unchecked components where the sign of the transitive predictions provides us with additional evidence, beyond the predictions of the pairwise matching model $f_{\theta TC}$, of the correctness of the component.

Finally, note that TransClean may not consume all of the initially predefined labeling budget $LB^{Total}$ if the amount of unlabeled removed edges predicted as a Match that create new negative transitive predictions is smaller than $LB^{EdgeRecovery}$. This is most likely when combining TransClean with robust pairwise matching models that easily reach Transitive Consistency on the initial stages of TransClean and do not flip their predictions on many removed unlabeled candidate edges after being finetuned.

### 5) LLM Labeling

When running TransClean, several steps of Algorithms 1, 2 and 3 involve the labeling of record pairs. This can be done either manually or using the pseudo-labels produced by a different model than the one used to produce the matching. In the experiments, we use an LLM for the labeling involved in the Initial TransClean Steps with Finetuning and the Post Finetuning Cleanup & Checks (Algorithms 1, 2) but not in the Edge Recovery step (Algorithm 3) due to the slower inference time of LLMs and the large number of record pairs requiring labels in this final step.

A drop in performance can be expected from using pseudo-labels since some true positives may be wrongly labeled as non matches and equivalently with false positives being labeled as true matches. In the experiments we compare the performance of TransClean using LLM pseudo-labels vs manual labeling and illustrate its robustness in effectively

**Algorithm 3** Edge Recovery

**Input:** Pairwise matching model $f_{\theta TC}$, matching graph $G^{TC}$, removed matches $\mathcal{R} = \{(r_i, r_j)_{rem} | f_{\theta TC}((r_i, r_j)_{rem}) = \texttt{Match}\}$, labeling budget $LB^{EdgeRecovery}$

**Output:** Final matching $G^{TC}$

**for** $(r_i, r_j)_{rem} \in \mathcal{R}$ **do** # *Check for each removed edge, which component it would belong to/create if added back*
  $c_i \leftarrow getComponent(r_i, G^{TC})$ #
  $c_j \leftarrow getComponent(r_j, G^{TC})$ #
  **if** $c_i == c_j$ **then**
    **Continue** # *This edge is already in the matching (as a transitive match) since both records already belong to the same component*
  **else**
    $Preds \leftarrow EvalNewTransMatches((r_i, r_j), G^{TC}, f_{\theta TC})$ # *Evaluate the transitive matches that adding this record back would create*
    **if** $Preds = \emptyset$ and $LB^{EdgeRecovery} > 0$ **then**
      # *Edge connects only a pair of records*
      $G^{TC} \leftarrow ManualCheck((r_i, r_j), G^{TC})$ # *Manually check the pair. If it is a match, add it back to the matching*
      $LB^{EdgeRecovery} \leftarrow LB^{EdgeRecovery} - 1$
    **else if** $pred == \texttt{Match}\ \forall\ pred \in Preds$ **then**:
      # *This edge connects two different components into a transitively consistent component*
      $G^{TC} \leftarrow AddBack((r_i, r_j), G^{TC})$ # *Add the pair back to the matching*
    **else if** $LB^{EdgeRecovery} > 0$ **then**:
      # *This edge produces a connected component that is not transitively consistent*
      $G^{TC} \leftarrow ManualCheck((r_i, r_j), G^{TC})$ # *Manually check the pair. If it is a match, add it back to the matching*
      $LB^{EdgeRecovery} \leftarrow LB^{EdgeRecovery} - 1$
**Return:** $G^{TC}$

detecting false positives even when the LLM produces the wrong labels.

### 6) TransClean Visual Example

Figure 6 provides a visualization of all the steps of TransClean involved in cleaning up the component presented in Figure 2. TransClean starts off by applying three *Initial Steps with Finetuning* with the goal to detect false positive edges via Minimum Edge Cuts and shortest paths (see edges ($r_{\#3}$, $r_{\#6}$), ($r_{\#4}$, $r_{\#6}$) in Finetuning Iter 1 and edges ($r_{\#2}$, $r_{\#9}$), ($r_{\#4}$, $r_{\#9}$) in Finetuning Iter 2 with orange dotted lines) to further finetune the pairwise matching model. Each of these edges is labeled, consuming the labeling budget, and consequently removed from the component (since they are false positives). Additionally, the Finetuning Iter 3 pruning step, which is done without labeling, removes 2 true positive edges (see the edges $r_{\#2}$, $r_{\#4}$) and ($r_{\#3}$, $r_{\#4}$)), since they form a minimum edge cut of the component. This is an example of removing a true positive match which needs to be recovered in the final edge recovery stage. These steps iteratively break up the component.

The last false positive edges ($r_{\#1}$, $r_{\#5}$) and ($r_{\#2}$, $r_{\#5}$) are pruned from the component during the *Post Finetuning Cleanup and Checks* and the removed true positive edges $\{(r_{\#i}, r_{\#4})\}_{i=1}^{3}$ are recovered during the *Edge Recovery* step.

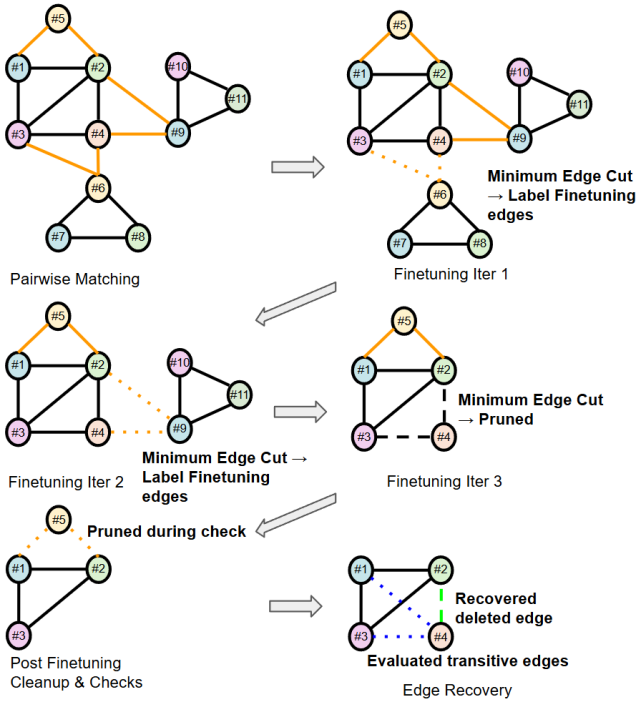


**FIGURE 6.** Evolution of the component of Figure 2 over the different steps of TransClean. Black edges illustrate true positive edges, orange edges false positive edges and dotted edges denote the selected edges, either for labeling or pruning, during a specific TransClean step.

## V. EXPERIMENTS

In this section we perform several end-to-end matching experiments with TransClean to address the following research questions:

- *How can TransClean be leveraged to provide an estimation of the quality of a given matching?*
- *Is TransClean able to improve matchings by detecting most of their false positive matches?*
- *How dependent is the performance of TransClean on the quality of the pairwise matching model it is combined with?*
- *What is the performance gap between manually labeling the candidate pairs selected by TransClean and pseudo-labelling them with an LLM?*

All our experiments were run on an Ubuntu machine with a single Nvidia Tesla T4 GPU, 16 VCPUs, and 64 GB of RAM.

### A. EXPERIMENTS SETUP

#### 1) Datasets

We experiment with the following multi-source datasets:

- **Synthetic Companies**: Presented in [12], this is a procedurally generated benchmark explicitly designed to mimic real-world matching conditions containing 868K records To the best of our knowledge, this is currently the largest entity matching data set used in the literature. Fixed training, validation, and test splits are provided. Additionally, it includes a set of 7K positive train pairs (involving 4K records) obtained via labeling the first 10K pairs through a heuristic blocking[7].
- **MusicBrainz**: Presented in [42], this benchmark is based on song records from the MusicBrainz database and uses a data generator to create duplicates with modified attribute values. We experiment on the 20k variant[8].
- **Camera/Monitor**: Presented in [10], these benchmarks consist of web-scraped JSON-files, each with different properties, containing product information from 24 and 26 different websites respectively. For both datasets, we cast all records into a table format by recording the `page_id` and merge all other attributes into a `text` attribute.
- **WDC Products**: Presented in [38], this benchmark consists on product data retrieved from 3,259 e-shops. Multiple variants of the dataset are available, varying in terms of the amount of corner-cases, unseen entities and development set size. We experiment on the variant with $80\%$ corner cases and a test set with $100\%$ unseen entities. Fixed training, validation, and test splits are provided.

See Table 1 for the general statistics of each of the datasets. For the MusicBrainz and Camera/Monitor datasets we implement our own train/val/test splits. We split based on a 60/20/20 proportion across the record groups, leading to differently sized splits (record groups have different numbers of records). This splitting makes sure that no test records are used during training. Unpaired records in the ground truth of the MusicBrainz dataset do not get assigned to any split.

#### 2) Machine Learning Models

We employ 2 different pairwise matching ML models in our experiments. DistilBERT [44], a distilled version of BERT, and CLER [53], a larger[9] state-of-the-art methodology explicitly developed for EM. We train both models as follows:

- **DistilBERT:** We fine-tune DistilBERT for 5 epochs and select the best performing epoch on the validation split. In case the dataset does not include negative pairs, we add them randomly in a proportion of 5 to 1.
- **CLER:** We train CLER with different labeling budgets[10]. As validation sets (used in CLER's training logic) we choose 500 randomly selected train pairs, as done in CLER's original implementation.

We do not use Large Language Models (LLMs) for pairwise matching due to their comparatively slower inference times. Rather, we use *DeepSeek LLM 7B Base* for labeling the candidate pairs selected by TransClean in the runs without manual labeling. We carry out this labeling by prompting the model with the following prompt:

"Do these two records represent the same entity? Answer only Yes or No, do not elaborate further. First record: [$record_1$] Second Record: [$record_2$]

Where $|record_i|$ denotes the representation of each record as a *string* via concatenating the *name/value* pairs of its attributes. We recover the labels from the LLM output via a regex, where we consider an output to be a `Match` prediction if it contains the substring *"yes"* and a `NoMatch` prediction otherwise.

#### 3) Blocking

Evaluating all possible record pairs in the test sets would be prohibitive[11] hence we need a blocking approach to reduce the amount of records that need to be evaluated. For DistilBERT we use a simple token overlap heuristic that pairs each record with the top 10 records ranked by the number of shared tokens between their token sequences. On the Synthetic Companies dataset we also use the token overlap heuristic used in [12]. CLER on the other hand, produces both an embedding-based `blocker` and a pairwise `matcher`. We use the `matcher` to evaluate the top 10 most similar pairs produced by the `blocker`. See Table 2 for a summary of the sets of candidate pairs used for each model along with their recall scores (the percentage of total true positives included in each set of pairs).

[7]We use this subset of labeled pairs during training, as being able to label all training pairs would be unrealistic in a real-world setting.

[8]All the differently sized variants can be retrieved from https://dbs.uni-leipzig.de/research/projects/benchmark-datasets-for-entity-resolution

[9]CLER uses RoBERTa as a classifier, which has 355M parameters compared to the 66M of DistilBERT.

[10]CLER trains via an uncertainty based active learning technique. As such, it requires an initial labeling budget to determine the number of training pairs to select. We use 10K for the Synthetic Companies, 5K for MusicBrainz & Camera, 2K for Monitor and 8K for WDC.

[11]For the Synthetic Companies dataset the number of records in the test set is 174K so we would need to evaluate (174K) * (174K - 1)/2 = 1.5 *$10^{10}$ record pairs.

**TABLE 1.** General dataset statistics. Highlighted in bold are the number of records we use during model training and the matching experiments.

| Dataset | Synthetic Companies | MusicBrainz | Camera | Monitor | WDC Products |
|---|---|---|---|---|---|
| **# of data sources** | 5 | 5 | 24 | 26 | 3,259 |
| **# of records** | 868K | 19K | 3.9K | 2.3K | 4.8K |
| **# of record groups** | 200K | 10K | 103 | 252 | 2.2K |
| **# of matches** | 1.5M | 16.25K | 157K | 13.5K | 28.3K |
| **# of records in** `train/val/test` | 520K**(4K)/174K/174K** | **14K/1.8K/456** | **2.5K/696/632** | **1.5K/570/196** | **2.8K/1K/1K** |
| **Attributes** | *name, city, region, country_code, short_description* | *number, length, artist, year, language* | *multiple (.json format)* | *multiple (.json format)* | *brand, title, description, price, priceCurrency* |

**TABLE 2.** Number of test candidate pairs considered by each of the models and their associated recall score.

| Dataset | Model | # of test candidate pairs & recall |
|---|---|---|
| **Synthetic Companies** | **DistilBERT** | 1.44M - 96.10% |
| | **CLER** | 430K - 98.25% |
| **MusicBrainz** | **DistilBERT** | 6.5K - 99.34% |
| | **CLER** | N/A[12] |
| **Camera** | **DistilBERT** | 5.9K - 83.38% |
| | **CLER** | 2.1K - 68.74% |
| **Monitor** | **DistilBERT** | 1.9K - 95.31% |
| | **CLER** | 519 - 88.02% |
| **WDC Products** | **DistilBERT** | 9.2K - 81.8% |
| | **CLER** | 3.6K - 62.6% |

#### 4) TransClean

We run TransClean on the matchings produced by the pairwise predictions of both models as illustrated in Figure 3. We set the number of initial steps with finetuning to $n = 5$ and the component size threshold to $S = 50$. We allocate $\frac{LB^{Total}}{2}$ to the Initial TransClean steps with finetuning and the remaining budget to the following steps. See Table 3 for a description of each of the parameters of TransClean.

**TABLE 3.** Description of TransClean parameters.

| Parameter | Notation | Explanation |
|---|---|---|
| Number of Initial Steps with Finetuning | $n$ | Number of repetitions of Algorithm 1 to carry out |
| Component Size Threshold | $S$ | Maximum size of a component to calculate its Minimum Edge Cut |
| Total Labeling Budget | $LB^{Total}$ | Total labeling budget to distribute across $LB^{init}$, $LB^{PostFT}$ and $LB^{EdgeRecovery}$, each corresponding to a step of TransClean |

### B. PAIRWISE MATCHING VS TRANSCLEAN

#### 1) Pairwise Matching

We start off our discussion by highlighting the challenges associated with attempting to match the Synthetic Companies dataset through pairwise matching only. A standard attempt would start by training CLER with a 10K labeling budget and the records involved in the labeled pairs described in Section V-A1. This will produce 2 models, a `blocker` and a `matcher`. The `blocker` produces a set of 430K candidate pairs, as noted in Table 2, of which the `matcher` considers 213K as matches. The question now is how to evaluate the quality of the produced matching. Blindly trusting the predictions of the `matcher` is risky, due to the likelihood of false positive matches. On the other extreme, manually checking all of the produced matches would require an impossibly large human effort. It is then natural to consider the transitive matches, which are implied by the pairwise predictions but haven not been evaluated by the model yet.

#### 2) Transitive Matches

As depicted in Figure 7, the 213K predicted matching pairs of CLER lead to 69K transitive matches of which 20.5K are predicted as matches and 48.5K as non-matches. This high number of negatively predicted transitive matches indicates that the `matcher` is not transitively consistent and that our matching likely has a considerable number of false positive matches.

#### 3) Running TransClean

In order to efficiently find the false positive matches, we execute TransClean with a 10K labelling budget as specified in Section V-A4 on the matching produced by the 213K predicted matching pairs. Figure 8 illustrates the evolution of the most relevant metrics of the matching during the different TransClean stages.

We start off labeling 1K record pairs in the first Finetuning Iteration and iteratively label up to 10K pairs through the different TransClean steps (see green bars). Notice how the number of negative transitive predictions (see purple bars) steadily decreases[13] whereas the number of positive transitive predictions (see red bars) largely remains constant. These two quantities indicate the gradual success of TransClean in pro-

[12]On the MusicBrainz dataset CLER's training failed to conclude after 40 hours of continuous training, so we omit it from our analysis.

[13]Except for the last Post Edge Recovery step, where the manually labeled pairs add transitive pairs that are true matches but are incorrectly predicted by the `matcher`.

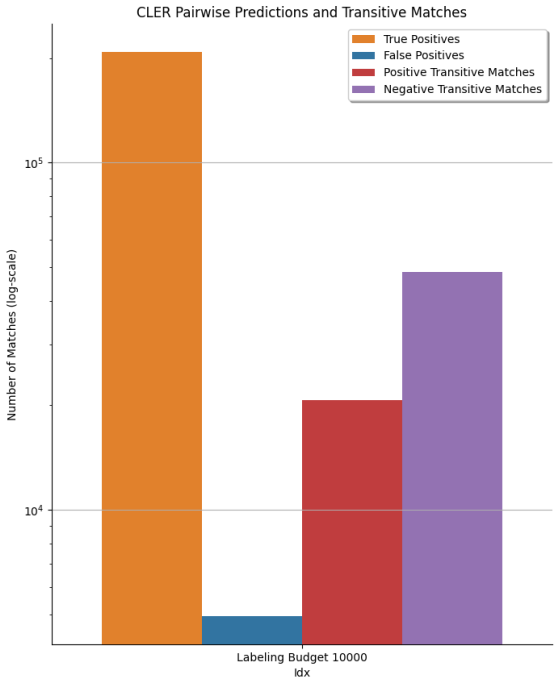


**FIGURE 7.** Aggregate pairwise predictions on the Synthetic Companies dataset produced by CLER trained with a labeling budget of 10K pairs. Even though the `matcher` model has a high precision, the number of negative transitive matches indicates a lack of Transitive Consistency and thus the likely presence of false positive matches.

gressively modifying the matching towards being transitively consistent while not removing too many of the initial true matches.

Note also the *correlations of both quantities with the amounts of true and false positives*, which makes them an ideal pair of proxies. In other words, positive transitive predictions are correlated with true positives, and negative transitive predictions are correlated with false positives. While the amounts of true and false positives are unobtainable in practice (since they would require manually labeling the entire matching), the numbers of positive and negative transitive predictions can be easily obtained, and are in fact calculated at each step of TransClean, simply by evaluating the transitive matches.

### C. RESULTS FOR ALL DATASETS

Table 4 presents the scores achieved by running TransClean, with manual labeling and LLM pseudo-labels, on the matching produced by each pairwise matching model for all datasets. The pairwise matching scores reflect the scores achieved considering only pairwise predictions. However, the Pre & Post TransClean scores incorporate the transitive matches, which makes them a better indicator of the quality of the matchings, as discussed in Section I.

#### 1) Pairwise Matching Results

We start our discussion with the *Pairwise Matching* and *Pre TransClean* scores. Both DistilBERT & CLER achieve in the Synthetic Companies dataset high F1 scores for pairwise matching ranging from 81.54 to 86.61. However, notice that with the inclusion of transitive matches the F1 score for DistilBERT drops to 0.04. This is expected, since transitive matches are true matches if they connect records from the same ground truth component (which increases the recall of the matching) but are non matches (i.e. counted as false positives and thus decrease the precision) otherwise. In the case of DistilBERT, a high pairwise F1 score of 81.54 hides the large number of erroneous transitive matches implied by the false positives the model produces[14], which make the *Pre TransClean* precision drop to 0.02. For CLER the inclusion of transitive matches has a positive impact on the Pre TransClean F1 score, which increases from 86.61 to 87.17 since the decrease in precision caused by adding the transitive matches is counteracted by the increase in recall. These results indicate that CLER's matcher is more robust than DistilBERT.

In the MusicBrainz dataset, DistilBERT achieves again high precision and recall, this time with a much smaller drop in terms of precision and F1 scores when adding into the matching the transitive matches. CLER on the other hand fails to conclude its training process after $40h$ of continuous training.

In the Camera and Monitor datasets both pairwise matching models yield matchings with very low recall. This is surprising, given the close-to-perfect scores reported in [54] and [53] but both of these works only experiment with a subset of the records (since no fixed splits are provided). [54] only compares records against those of the same category and [53] uses only 2 data sources out of the 24 included in the monitor dataset. In turn, we use all labeled records from both datasets. The low F1 scores achieved by both models signify that the actual difficulty posed by these datasets has been largely sidestepped in previous works through downsampling.

Finally in the WDC Products dataset, DistilBERT produces a matching with low pairwise precision and consequently also very low Pre TransClean precision. CLER on the other hand, produces a model with extremely low recall but perfect precision, similarly as in the Camera and Monitor datasets. This signifies that CLER's uncertainty based training is not successful in datasets with a lot of diversity and inherently difficult records as it produces models with very low recall.

[14]These false positives lead to far more and far larger connected components, which imply the large number of erroneous transitive matches.

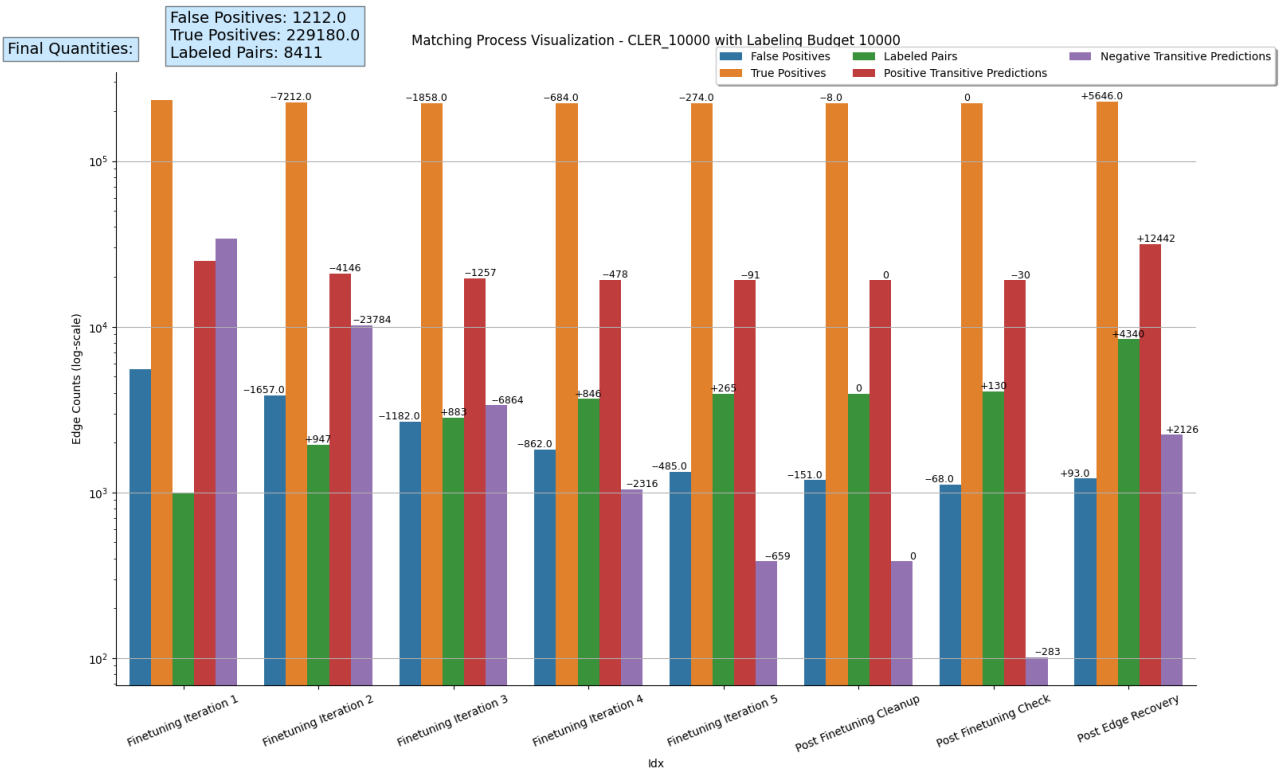


**FIGURE 8.** Visualization of relevant metrics of a matching on the Synthetic Companies dataset and their evolution over the different steps of TransClean. Each set of bars illustrates from left to right the number of: 1. False Positives 2. True Positives 3. Labeled Pairs 4. Positive Transitive Predictions 5. Negative Transitive Predictions with their variations w.r.t to the previous column from the second Finetuning Iteration 2 onward. The amounts of true and false positives include all the matches present in the matching, both predicted and transitive.

### 2) TransClean Results

TransClean improves the matchings produced by all pairwise models[15], with an average F1 score improvement of +24.42 between Pre and Post TransClean scores when run with manual labeling, illustrating the importance of removing false positives from matchings. Across all datasets, TransClean manages to consistently remove a substantial proportion of false positives from the matchings produced by both models. However, when run with DistilBERT it removes a much higher percentage of true positives than when run with CLER because the former produces a lot more false positives and thus needs to remove a lot more pairs from the matchings in order to reach Transitive Consistency.

When combined with an effective pairwise matching model, as is the case of both models in the MusicBrainz dataset, DistilBERT in the WDC products and CLER in the Synthetic Companies, TransClean removes very few true positives, leading to a substantial improvement in F1 scores. This illustrates how TransClean can detect most of the false positives of well performing models.

When using an LLM to produce pseudo-labels, we can observe that the percentage of removed true positives increases while that of the removed false positives largely decreases across all datasets. However the differences in the datasets where the pairwise matching models achieve good results (Synthetic Companies and MusicBrainz) are small, due to both the LLM performance and TransClean's tolerance to pseudo-labels. In the other datasets, labeling with the LLM leads to removing large proportions of the original true positives, both because the matchings start with very few matches due to the low recall and because in these datasets the transitive matches are not a good indication of the matching's quality due to the poor performance of the pairwise matching models.

In terms of running times, training DistilBERT as we describe in Section V-A2 is faster than training CLER, whose uncertainty based active learning training leads to much longer training times. Inference times are similar for both models, given that they both have a relatively low number of parameters for today's standards, with CLER being slower than DistilBERT due to its `matcher` having 5 times more parameters.TransClean running times when doing manual

[15]Except for the CLER on the Musicbrainz dataset where it fails to conclude its training process and on the Monitor and WDC Products datasets since the matches it produces have perfect precision but extremely low recall. Since these matchings have no false positives, TransClean cannot be applied.

**TABLE 4.** Scores achieved by each combination of pairwise classifier and TransClean settings (labeling budget vs LLM labeling). Ditto marks indicate the repetition of the value above in the respective column (experiments that use the same pairwise matching model will have the same Pairwise Matching and Pre TransClean scores and Training/Inference times). Note that the labeling budget and the LLM generated labeling (see column "TransClean Variant") are only used during the TransClean phase.

| Dataset | Pairwise Matching Model | TransClean Variant | Pairwise Matching (pairs from blocking) | | | Pre TransClean (including transitive matches) | | | Post TransClean (including transitive matches) | | | | | Running Times hh:mm:ss | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Precision | Recall | F1 Score | Precision | Recall | F1 Score | % True Positives Removed | % False Positives Removed | Precision | Recall | F1 Score | Training Time | Inference Time | TransClean Time |
| **Synthetic Companies** | **DistilBERT** | 10K budget | 85.86 | 77.63 | 81.54 | 0.02 | 83.68 | 0.04 | 36.35% | 96.07% | 94.7 | 54.85 | 69.47 | 11:32:20 | 01:09:24 | 02:36:44 |
| | **DistilBERT** | LLM Labeling | " | " | " | " | " | " | 42.00% | 96.67% | 97.61 | 47.71 | 64.09 | " | " | 27:51:32 |
| | **CLER** | 10K budget | 97.69 | 77.79 | 86.61 | 87.48 | 86.87 | 87.17 | 1.88% | 78.06% | 98.54 | 86.67 | 92.22 | 14:49:57 | 01:36:37 | 01:18:42 |
| | **CLER** | LLM Labeling | " | " | " | " | " | " | 3.52% | 72.06% | 99.02 | 82.83 | 90.2 | " | " | 18:00:15 |
| **MusicBrainz** | **DistilBERT** | 1K budget | 94.74 | 98.68 | 96.67 | 86.04 | 98.68 | 91.93 | 0% | 56% | 97.61 | 98.68 | 98.14 | 01:16:16 | 00:00:26 | 00:00:12 |
| | **DistilBERT** | LLM Labeling | " | " | " | " | " | " | 3.11% | 48.0% | 96.9 | 95.83 | 96.36 | " | " | 00:09:15 |
| | **CLER** | *N/A* | – | – | – | – | – | – | – | – | – | – | – | *Out of time* | – | – |
| **Camera** | **DistilBERT** | 1K budget | 86.00 | 2.44 | 4.75 | 66.88 | 6.49 | 11.83 | 0% | 76.19% | 96.8 | 6.49 | 12.16 | 11:28:32 | 00:00:55 | 00:00:32 |
| | **DistilBERT** | LLM Labeling | " | " | " | " | " | " | 87.59% | 73.02% | 55.81 | 0.3 | 0.6 | " | " | 00:27:17 |
| | **CLER** | 1K budget | 98.5 | 0.83 | 1.65 | 97.27 | 1.12 | 2.21 | 0% | 50% | 99.44 | 1.12 | 2.22 | 16:19:32 | 00:07:30 | 00:08:07 |
| | **CLER** | LLM Labeling | " | " | " | " | " | " | 35.88% | 50% | 98.84 | 0.54 | 1.07 | " | " | 00:10:31 |
| **Monitor** | **DistilBERT** | 1K budget | 36.12 | 23.79 | 28.69 | 37.98 | 38.35 | 38.16 | 8.16% | 57.31% | 60.19 | 31.55 | 41.4 | 00:55:39 | 00:00:10 | 00:00:55 |
| | **DistilBERT** | LLM Labeling | " | " | " | " | " | " | 30.61% | 13.46% | 35.89 | 27.99 | 31.45 | " | " | 00:38:42 |
| | **CLER** | *N/A* | 100 | 14.72 | 25.66 | 100 | 14.72 | 25.66 | – | – | – | – | – | 09:46:01 | 00:02:57 | – |
| **WDC Products** | **DistilBERT** | 10K budget | 34.52 | 63.8 | 44.8 | 1.13 | 66.6 | 2.22 | 12.22% | 86.77% | 75.54 | 56.2 | 64.45 | 02:14:00 | 00:00:40 | 00:02:12 |
| | **DistilBERT** | LLM Labeling | " | " | " | " | " | " | 46.7% | 77.52% | 49.86 | 34.6 | 40.85 | " | " | 01:36:32 |
| | **CLER** | *N/A* | 100 | 0.2 | 0.4 | 100 | 0.2 | 0.4 | – | – | – | – | – | 14:39:32 | 00:10:10 | – |

labeling are comparable to the inference times across all datasets. This is because running TransClean involves doing inference on a similar number of record pairs and finetuning with very small amount of labeled pairs. However, in these experiments manual labeling is simulated with a ground truth lookup and thus is not reflected in the corresponding times. For a fair comparison between manual and LLM labeling, an additional labeling time should be considered. Additionally, the runs with LLM labeling are much slower due to the additional time it takes to do LLM inference to produce pseudo-labels as well as the fact that the pseudo-labels produce comparatively more transitive matches due to the possibility of labeling false positives as true positives. In terms of comparing DistilBERT and CLER, the matchings produced by DistilBERT have a lot more false positives and thus TransClean takes longer to run.

## VI. CONCLUSIONS & FUTURE WORK

In this work, we have presented TransClean, a method that leverages the Transitive Consistency of a pairwise model to gradually remove false positive matches while removing few true matches. We have shown how the proposed method can be used to remove false positives from matches produced by pairwise matching methods trained naively (DistilBERT) as well as state-of-the-art matchers (CLER) and analyzed the differences in its performance when using an LLM to produce pseudo-labels compared to doing manual labeling.

We have shown how the positive and negative transitive predictions calculated during the different stages of TransClean can be used as proxies for the true and false positives of a matching as well as leveraged to detect false positives if the pairwise matching model used to produce the matching is effective. Our experiments show that, in terms of the actual record groups produced in multi-source matching settings, the effect of even small quantities of false positives should not be ignored. However, TransClean is able to detect such model mistakes if it is combined with a well performing pairwise matching model.

Beyond entity matching, the principles of Transitive Consistency may generalize to domains like knowledge graph refinement, author disambiguation, or anomaly detection, where consistency across relational structures is key. We plan to explore these applications in future work.

## ACKNOWLEDGEMENTS

We thank Andrea Nagy and Barnabas Gera, both formerly of Move Digital AG, for their valuable insight during the course of the project. The work was funded by Innosuisse as an innovation project under the project number `54383.1 IP-ICT`.

## REFERENCES

[1] Y. Altowim, D. V. Kalashnikov, and S. Mehrotra. Progresser: Adaptive progressive approach to relational entity resolution. ACM Trans. Knowl. Discov. Data, 12(3), Mar. 2018.

[2] N. Barlaug. Tailoring entity matching for industrial settings. In Proceedings of the 29th ACM International Conference on Information & Knowledge Management, CIKM '20, page 3217–3220, New York, NY, USA, 2020. Association for Computing Machinery.

[3] I. Bhattacharya and L. Getoor. Collective entity resolution in relational data. ACM Transactions on Knowledge Discovery from Data (TKDD), 1(1):5–es, 2007.

[4] L. Bischof, S. Teodoropol, R. M. Füchslin, and K. Stockinger. Hybrid quantum neural networks show strongly reduced need for free parameters in entity matching. Scientific Reports, 15(1):4318, 2025.

[5] U. Brunner and K. Stockinger. Entity matching on unstructured data: an active learning approach. In 2019 6th Swiss Conference on Data Science (SDS), pages 97–102. IEEE, 2019.

[6] U. Brunner and K. Stockinger. Entity matching with transformer architectures - a step forward in data integration. 23rd International Conference on Extending Database Technology, Copenhagen, 30 March - 2 April 2020, 2020.

[7] Z. Chen, Q. Chen, B. Hou, M. Ahmed, and Z. Li. Improving Machine-based Entity Resolution with Limited Human Effort: A Risk Perspective. arXiv e-prints, page arXiv:1805.12502, May 2018.

[8] Z. Chen, Q. Chen, B. Hou, T. Duan, Z. Li, and G. Li. Towards Interpretable and Learnable Risk Analysis for Entity Resolution. arXiv e-prints, page arXiv:1912.02947, Dec. 2019.

[9] V. Christophides, V. Efthymiou, T. Palpanas, G. Papadakis, and K. Stefanidis. An overview of end-to-end entity resolution for big data. ACM Comput. Surv., 53(6), Dec. 2020.

[10] V. Crescenzi, A. De Angelis, D. Firmani, M. Mazzei, P. Merialdo, F. Piai, and D. Srivastava. Alaska: A Flexible Benchmark for Data Integration Tasks. arXiv e-prints, page arXiv:2101.11259, Jan. 2021.

[11] S. Das, A. Doan, P. S. G. C., C. Gokhale, P. Konda, Y. Govind, and D. Paulsen. The magellan data repository. https://sites.google.com/site/anhaidgroup/projects/data.

[12] F. de Meer Pardo, C. Lehmann, D. Gehrig, A. Nagy, S. Nicoli, B. H. Misheva, M. Braschler, and K. Stockinger. Gralmatch: Matching groups of entities with graphs and language models. In A. Simitsis, B. Kemme, A. Queralt, O. Romero, and P. Jovanovic, editors, Proceedings 28th International Conference on Extending Database Technology, EDBT 2025, Barcelona, Spain, March 25-28, 2025, pages 1–12. OpenProceedings.org, 2025.

[13] H. L. Dunn. Record linkage. Am. J. Public Health Nations. Health, 36(12):1412–1416, Dec. 1946.

[14] M. Ebraheem, S. Thirumuruganathan, S. Joty, M. Ouzzani, and N. Tang. Distributed representations of tuples for entity resolution. Proc. VLDB Endow., 11(11):1454–1467, July 2018.

[15] I. P. Fellegi and A. B. Sunter. A theory for record linkage. Journal of the American Statistical Association, 64:1183–1210, 1969.

[16] C. Fu, X. Han, J. He, and L. Sun. Hierarchical matching network for heterogeneous entity resolution. In C. Bessiere, editor, Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20, pages 3665–3671. International Joint Conferences on Artificial Intelligence Organization, 7 2020. Main track.

[17] C. Fu, X. Han, L. Sun, B. Chen, W. Zhang, S. Wu, and H. Kong. End-to-end multi-perspective matching for entity resolution. In International Joint Conference on Artificial Intelligence, 2019.

[18] C. Ge, P. Wang, L. Chen, X. Liu, B. Zheng, and Y. Gao. Collaborem: A self-supervised entity matching framework using multi-features collaboration. IEEE Transactions on Knowledge and Data Engineering, 35(12):12139–12152, 2023.

[19] B. Genossar, A. Gal, and R. Shraga. The Battleship Approach to the Low Resource Entity Matching Problem. arXiv e-prints, page arXiv:2311.15685, Nov. 2023.

[20] O. Hassanzadeh, F. Chiang, R. J. Miller, and H. C. Lee. Framework for evaluating clustering algorithms in duplicate detection. Proc. VLDB Endow., 2:1282–1293, 2009.

[21] B. Hou, Q. Chen, Z. Chen, Y. Nafa, and Z. Li. r-humo: A risk-aware human-machine cooperation framework for entity resolution with quality guarantees. IEEE Transactions on Knowledge and Data Engineering, 32(2):347–359, 2020.

[22] D. V. Kalashnikov and S. Mehrotra. Domain-independent data cleaning via analysis of entity-relationship graph. ACM Trans. Database Syst., 31(2):716–767, June 2006.

[23] D. R. Karger. Minimum Cuts in Near-Linear Time. arXiv e-prints, page cs/9812007, Dec. 1998.

[24] J. Kasai, K. Qian, S. Gurajada, Y. Li, and L. Popa. Low-resource Deep Entity Resolution with Transfer and Active Learning. arXiv e-prints, page arXiv:1906.08042, June 2019.

[25] P. Konda, S. Das, P. Suganthan G. C., A. Doan, A. Ardalan, J. R. Ballard, H. Li, F. Panahi, H. Zhang, J. Naughton, S. Prasad, G. Krishnan, R. Deep, and V. Raghavendra. Magellan: Toward building entity matching management systems. Proc. VLDB Endow., 9(12):1197–1208, aug 2016.

[26] H. Köpcke, A. Thor, and E. Rahm. Evaluation of entity resolution approaches on real-world match problems. Proc. VLDB Endow., 3(1–2):484–493, Sept. 2010.

[27] P. Kouki, J. Pujara, C. Marcum, L. Koehly, and L. Getoor. Collective entity resolution in familial networks. In 2017 IEEE International Conference on Data Mining (ICDM), pages 227–236, 2017.

[28] B. Li, W. Wang, Y. Sun, L. Zhang, M. A. Ali, and Y. Wang. Grapher: Token-centric entity resolution with graph convolutional neural networks. Proceedings of the AAAI Conference on Artificial Intelligence, 34(05):8172–8179, Apr. 2020.

[29] Y. Li, J. Li, Y. Suhara, A. H. Doan, and W. C. Tan. Effective entity matching with transformers. VLDB Journal, pages 1–21, 1 2023.

[30] Y. Li, J. Li, Y. Suhara, J. Wang, W. Hirota, and W.-C. Tan. Deep entity matching: Challenges and opportunities. J. Data and Information Quality, 13(1), jan 2021.

[31] J. Maciejewski, K. Nikoletos, G. Papadakis, and Y. Velegrakis. Progressive entity matching: A design space exploration. Proc. ACM Manag. Data, 3(1), Feb. 2025.

[32] Z. Miao, Y. Li, and X. Wang. Rotom: A meta-learned data augmentation framework for entity matching, data cleaning, text classification, and beyond. In Proceedings of the 2021 International Conference on Management of Data, pages 1303–1316, 2021.

[33] A. Narayan, I. Chami, L. Orr, S. Arora, and C. Ré. Can Foundation Models Wrangle Your Data? arXiv e-prints, page arXiv:2205.09911, May 2022.

[34] M. Nentwig, A. Groß, and E. Rahm. Holistic entity clustering for linked data. In 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW), pages 194–201, 2016.

[35] R. Pandey, H. Purohit, C. Castillo, and V. L. Shalin. Modeling and mitigating human annotation errors to design efficient stream processing systems with human-in-the-loop machine learning. International Journal of Human-Computer Studies, 160:102772, 2022.

[36] G. Papadakis, G. Mandilaras, L. Gagliardelli, G. Simonini, E. Thanos, G. Giannakopoulos, S. Bergamaschi, T. Palpanas, and M. Koubarakis. Three-dimensional entity resolution with jedai. Information Systems, 93:101565, 05 2020.

[37] R. Peeters and C. Bizer. Supervised contrastive learning for product matching. Companion Proceedings of the Web Conference 2022, 2022.

[38] R. Peeters, R. Chiz Der, and C. Bizer. WDC Products: A Multi-Dimensional Entity Matching Benchmark. arXiv e-prints, page arXiv:2301.09521, Jan. 2023.

[39] R. Peeters, A. Steiner, and C. Bizer. Entity Matching using Large Language Models. arXiv e-prints, page arXiv:2310.11244, Oct. 2023.

[40] A. Primpeli and C. Bizer. Graph-boosted active learning for multi-source entity resolution. In The Semantic Web – ISWC 2021: 20th International Semantic Web Conference, ISWC 2021, Virtual Event, October 24–28, 2021, Proceedings, page 182–199, Berlin, Heidelberg, 2021. Springer-Verlag.

[41] A. Saeedi, M. Nentwig, E. Peukert, and E. Rahm. Scalable matching and clustering of entities with famer. Complex Systems Informatics and Modeling Quarterly, (16):61–83, Oct. 2018.

[42] A. Saeedi, E. Peukert, and E. Rahm. Comparative Evaluation of Distributed Clustering Schemes for Multi-source Entity Resolution, page 278–293. Springer International Publishing, 2017.

[43] A. Saeedi, E. Peukert, and E. Rahm. Using link features for entity clustering in knowledge graphs. In The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings, page 576–592, Berlin, Heidelberg, 2018. Springer-Verlag.

[44] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv e-prints, page arXiv:1910.01108, Oct. 2019.

[45] G. Simonini, L. Zecchini, S. Bergamaschi, and F. Naumann. Entity resolution on-demand. Proc. VLDB Endow., 15(7):1506–1518, Mar. 2022.

[46] S. Thirumuruganathan, H. Li, N. Tang, M. Ouzzani, Y. Govind, D. Paulsen, G. Fung, and A. Doan. Deep learning for blocking in entity matching: A design space exploration. Proc. VLDB Endow., 14(11):2459–2472, jul 2021.

[47] V. Vamsikrishna Meduri, L. Popa, P. Sen, and M. Sarwat. A Comprehensive Benchmark Framework for Active Learning Methods in Entity Matching. arXiv e-prints, page arXiv:2003.13114, Mar. 2020.

[48] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention Is All You Need. arXiv e-prints, page arXiv:1706.03762, June 2017.

[49] R. Wang, Y. Li, and J. Wang. Sudowoodo: Contrastive self-supervised learning for multi-purpose data integration and preparation, 2022.

[50] T. Wang, X. Chen, H. Lin, X. Chen, X. Han, H. Wang, Z. Zeng, and L. Sun. Match, Compare, or Select? An Investigation of Large Language Models for Entity Matching. arXiv e-prints, page arXiv:2405.16884, May 2024.

[51] S. E. Whang, D. Marmaros, and H. Garcia-Molina. Pay-as-you-go entity resolution. IEEE Transactions on Knowledge and Data Engineering, 25(5):1111–1124, 2013.

[52] R. Wrembel. Data integration, cleaning, and deduplication: Research versus industrial projects. In E. Pardede, P. Delir Haghighi, I. Khalil, and G. Kotsis, editors, Information Integration and Web Intelligence, pages 3–17, Cham, 2022. Springer Nature Switzerland.

[53] S. Wu, Q. Wu, H. Dong, W. Hua, and X. Zhou. Blocker and matcher can mutually benefit: A co-learning framework for low-resource entity resolution. Proceedings of the VLDB Endowment, 17:292–304, 01 2024.
[54] D. Yao, Y. Gu, G. Cong, H. Jin, and X. Lv. Entity resolution with hierarchical graph attention networks. Proceedings of the 2022 International Conference on Management of Data, 2022.
[55] W. Zhang, H. Wei, B. Sisman, X. Luna Dong, C. Faloutsos, and D. Page. AutoBlock: A Hands-off Blocking Framework for Entity Matching. arXiv e-prints, page arXiv:1912.03417, Dec. 2019.
[56] Z. Zhang, P. Groth, I. Calixto, and S. Schelter. A deep dive into cross-dataset entity matching with large and small language models, 2025.
[57] C. Zhao and Y. He. Auto-em: End-to-end fuzzy entity-matching using pre-trained deep models and transfer learning. In The World Wide Web Conference, WWW '19, page 2413–2424, New York, NY, USA, 2019. Association for Computing Machinery.

## AUTHOR BIOS

**Fernando de Meer Pardo** is a research scientist specialized in Financial Data Science with a focus on GenAI and NLP problems in Databases/Data Management. He has graduated from the PhD program in Data Science of the University of Zurich in May 2025.

**Branka Hadji Misheva** is a Professor of Applied Data Science and Finance at Bern University of Applied Sciences, Switzerland, where she leads the Applied AI Research and Solutions (AIRS) group. Branka has led multiple research projects funded by MSCA, Innosuisse, and SNF, focusing on AI applications in finance, credit risk modeling, and explainability in machine learning. She is an active contributor to international research networks, including COST FinAI. She has experience collaborating with industry partners, regulatory bodies, and central banks to develop responsible AI solutions. Her expertise spans deep learning, financial risk modeling, and counterfactual explanations. She is also an editor and reviewer for leading academic journals in AI and finance.

**Martin Braschler** is Director of the Institute of Computer Science at the Zurich University of Applied Sciences, Switzerland, and holds the title of Professor ZFH. His main research is in the field of information retrieval, including cross-language information retrieval and evaluation of search systems. Martin Braschler was a visiting scientist at National Institute of Standards and Technology (NIST), USA and Humboldt-University of Berlin, Germany. He has extensive experience in bringing search applications to the marketplace.

**Kurt Stockinger** is Professor of Computer Science, Director of Studies in Data Science, and Head of the Intelligent Information Systems Group at Zurich University of Applied Sciences, Switzerland. He is also affiliated with University of Zurich. His research interests are at the intersection of information systems, natural language processing and machine learning. Previously Kurt Stockinger was (1) a visiting scholar at University of Washington in Seattle, Washington, USA, (2) he worked at Credit Suisse in Zurich, Switzerland, (3) at Lawrence Berkeley National Laboratory in Berkeley, California, USA, (4) at California Institute of Technology in Pasadena, California, USA, (5) as well as at CERN in Geneva, Switzerland. He holds a Ph.D. in computer science from CERN / University of Vienna.