Graph Metrics-driven Record Cluster Repair meets LLM-based active learning

VICTOR CHRISTEN, Faculty of Mathematics and Computer Science, Leipzig University, Leipzig, Germany
DANIEL OBRACZKA, Faculty of Mathematics and Computer Science, Leipzig University, Leipzig, Germany
MARVIN HOFER, Faculty of Mathematics and Computer Science, Leipzig University, Leipzig, Germany
MARTIN FRANKE, Faculty of Mathematics and Computer Science, Leipzig University, Leipzig, Germany
ERHARD RAHM, Faculty of Mathematics and Computer Science, Leipzig University, Leipzig, Germany

Entity resolution plays an important role in data integration. However, most entity resolution methods focus on pairwise linkage and ignore potential errors generated by the transitive closure based on the determined equality links between two or more data sources. The transitive closure of a record forms a cluster where each record represents the same entity. Cluster repair methods aim to determine these errors and correct them. In the first category of methods, the assumption is that the data sources themselves do not contain any duplicates. Consequently, each cluster can contain at most one record from the same data source. However, real-world data often deviates from this assumption due to quality issues. Recent approaches apply clustering methods in combination with link categorization methods or graph clustering algorithms based on a single graph metric so they can be applied to data sources with duplicates. Nevertheless, the quality of the results highly varies depending on the configuration and dataset. In this study, we introduce a novel approach for cluster repair that utilizes graph metrics derived from the underlying similarity graphs. These metrics enable a comprehensive characterization of links and the generation of enhanced classification models. In addition to graph metric-based models, we integrate an active learning mechanism tailored to cluster-specific attributes. Moreover, we integrate large language models as an oracle. The evaluation shows that the graph metric-based method outperforms existing cluster repair methods and is more robust regarding different datasets and configurations.

CCS Concepts: • **Information systems → Entity resolution;**

Additional Key Words and Phrases: Record linkage, cluster repair, graph analysis, machine learning, active learning, LLM

This work was partially supported by the German Federal Ministry of Education and Research by funding the "Center for Scalable Data Analytics and Artificial Intelligence Dresden/Leipzig" (ScaDS.AI).
Authors' Contact Information: Victor Christen, Faculty of Mathematics and Computer Science, Leipzig University, Leipzig, Sachsen, Germany; e-mail: christen@informatik.uni-leipzig.de; Daniel Obraczka, Faculty of Mathematics and Computer Science, Leipzig University, Leipzig, Sachsen, Germany; e-mail: obraczka@informatik.uni-leipzig.de; Marvin Hofer, Faculty of Mathematics and Computer Science, Leipzig University, Leipzig, Sachsen, Germany; e-mail: hofer@informatik.uni-leipzig.de; Martin Franke, Faculty of Mathematics and Computer Science, Leipzig University, Leipzig, Sachsen, Germany; e-mail: Martin.Franke@gmx-topmail.de; Erhard Rahm, Faculty of Mathematics and Computer Science, Leipzig University, Leipzig, Sachsen, Germany; e-mail: rahm@informatik.uni-leipzig.de.

ACM 1936-1955/2025/06-ART7
https://doi.org/10.1145/3735511

ACM Reference Format:
Victor Christen, Daniel Obraczka, Marvin Hofer, Martin Franke, and Erhard Rahm. 2025. Graph Metrics-driven Record Cluster Repair meets LLM-based active learning. *ACM J. Data Inform. Quality* 17, 2, Article 7 (June 2025), 25 pages. https://doi.org/10.1145/3735511

1 Introduction

With the exponential growth of data, integrating information from multiple sources has become essential for enabling comprehensive analysis and decision-making. A key step in data integration beside schema mapping and data fusion is **entity resolution** (**ER**) [4]. Also known as record linkage, duplicate detection, or entity matching [12], ER identifies records representing the same entity across different data sources.

Over the past few decades, numerous ER methods [10, 12, 34] have been developed with most focusing on pairwise linkage between two data sources [8, 14, 25, 35, 46]. The integration of multiple data sources requires the execution of various pairwise linkage processes where the results are consolidated by computing the transitive closures also called clusters utilizing the generated record links. Each cluster is expected to consist of records corresponding to the same entity. The transitive closure can lead to the propagation of linkage errors and consequently to erroneous clusters where records are included representing different entities. To address this issue, cluster repair methods aim to refine clusters by correcting erroneous links [13, 20, 41, 44, 48]. The methods can be divided into two categories. The first [13, 36, 44, 48] assumes that data sources are free of duplicates. The methods of the second category can be applied to ER problems with data sources consisting of duplicates by themselves. Leveraging the duplicate-free assumption, the approach removes links and produces clusters that conform to this ideal condition. However, this assumption is often unrealistic in practice, as data sources in real-world applications frequently contain duplicates. As a result, methods that rely on duplicate-free assumptions tend to perform poorly when dealing with "dirty" data sources containing errors and inconsistencies.

Initial cluster repair methods [20] focus on a single data source without any duplicate-free assumptions. Hassanzadeh and colleague [20] utilized single-pass clustering algorithms. Saeedi et al. [44] compared their approach with a subset of the best algorithms from [20] and outperformed the single-pass clustering algorithms on duplicate-free data sources.

Recent approaches [27, 41, 43] of the second category have introduced modified clustering techniques that can repair clusters from noisy data sources as well as clean data sources. Despite these advancements, the effectiveness of such methods remains highly dependent on specific configurations tailored to particular linkage problems.

Our work focuses on cluster repair within the ER process, applicable to both clean-clean and dirty linkage tasks with minimal configuration effort. To overcome existing limitations, we propose *GraphCR*, a novel Graph metrics-driven Cluster Repair approach. GraphCR uses multiple graph metric-based features to identify correct and incorrect links within clusters. Table 1 provides an overview of the graph metrics used to build a classification model for predicting link correctness. By leveraging machine learning with diverse graph metric-based features, our method achieves accurate and robust cluster repair results, outperforming approaches that depend solely on single metrics or similarities.

Specifically, we make the following contributions:

— We propose a novel cluster repair method *GraphCR* that employs a classification model using a variety of graph metric-based features. In addition to the similarities, the used features cover network information within a cluster leading to a meaningful representation of links. The repair step utilizes the model to determine erroneous links from the clusters iteratively.

Table 1. Overview of Graph Metric-Based Features to Characterize an Edge

Name	Element type	Scope
PageRank [5]	node	network
closeness centrality [42]	node	network
betweenness centrality [17]	node	network
cluster coefficient [23]	node	network
normal link ratio [44]	node	network
strong link ratio [44]	node	network
similarity	edge	local
bridge [47]	edge	network
betweenness centrality [17]	edge	network
complete ratio	graph	network
weighted degree	node	local

We distinguish between the element type of a graph and the scope of information.

— We integrate an active learning approach into our method combined with a LLM-based oracle. To generate representative training data regarding the different clusters with their specific characteristic, we extend an existing active learning method by considering cluster-specific features in the selection phase. We also integrate the usage of LLMs as an oracle instead of a human-based one.
— We evaluate GraphCR on three datasets, analyzing labeling budgets, selection strategies, and the use of LLMs for training data generation. Results demonstrate that GraphCR is more robust and effective than existing methods across various datasets and configurations. Additionally, we validate its robustness against noisy similarity graphs by introducing random changes to edge similarities.

The remainder of this article is structured as follows. In Section 2, we define the problem of repairing clusters of records. Moreover, we discuss graph metrics as an integral component of our approach. In Section 3, we discuss existing cluster repair methods and additionally active learning and LLM-based ER methods. In Section 4, we present our novel approach for repairing clusters utilizing graph metric-based features with active learning. In Section 5, we evaluate our approach on different datasets to validate its practical applicability. Finally, we conclude our work in Section 6.

2 Preliminaries

2.1 Problem Definition

Let $\mathcal{D}$ be a set of data sources $D_1, D_2, \ldots, D_n$ consisting of records $r1, \ldots, rk$. Moreover, an arbitrary pairwise record linkage system generates a similarity graph $SG = (V, E)$. V represents the set of records across the different data sources $\mathcal{D}$ and E is the set of weighted edges connecting two records.

The edges are generated by the record linkage system or derived from a collection of links. A record linkage tool generates for each data source pair a set of links by executing the following steps: preprocessing, blocking, similarity computation, and classification [9].

We generate similarities for each pair of data sources to create a comprehensive similarity graph, which serves as a crucial element in our cluster repair methods. In contrast, incremental ER methods focus on comparing only the integrated data source with a single data source. However, research by Saeedi et al. [45] showed that the order of data sources significantly impacts the quality of the resolution process.

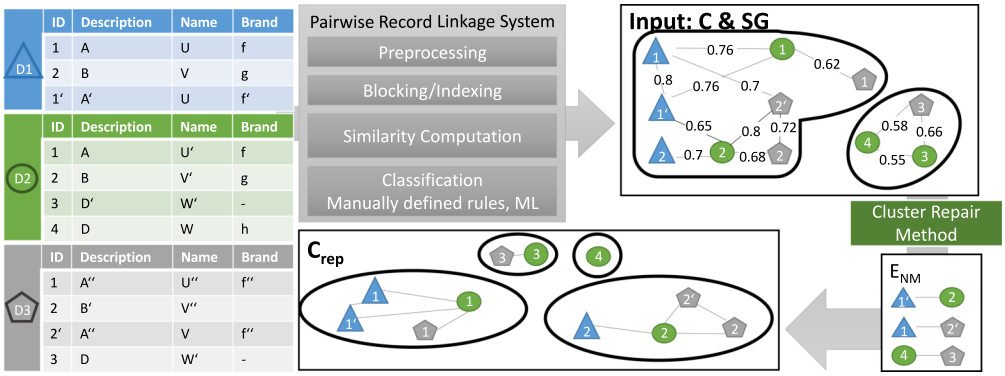


Fig. 1. Outline of the complete ER process including the repair method identifying incorrect edges $\mathbf{E_{NM}}$ to construct repaired clusters $\mathbf{C_{rep}}$. The different shapes of the records represent the data sources (D1: triangle, D2: circle, D3: pentagon).

In general, a record linkage approach computes multiple similarities regarding different attributes. In this work, we assume that a function aggregates the similarities. Moreover, a similarity can also be a probability of a classification model of how likely the match is. Therefore, an edge's weight is between 0 and 1. Note, that the similarity graph is incomplete because of the classification step, so not each record pair represents an edge. Due to the transitivity of equality links, the records of a connected component represent a cluster $c \in \mathbf{C}$ and, thus, the same entity. However, clusters may include incorrectly assigned records due to linkage errors.

Therefore, a *cluster repair method* [12] aims to determine incorrect clusters and repair them. The repair method (see Equation (1)) identifies incorrect edges $\mathbf{E_{NM}}$ utilizing the computed similarity graph $\mathbf{SG}$ with its similarities. The deletion of edges of $\mathbf{E_{NM}}$ in the similarity graph $\mathbf{SG}$ and the computation of the transitive closure lead to a repaired set of clusters $\mathbf{C_{rep}}$.

$$\mathbf{E_{NM}} \leftarrow cluster\_repair(\mathbf{SG}, \mathbf{C}) \tag{1}$$

$$\mathbf{C_{rep}} \leftarrow transitive\_closure(edges(\mathbf{SG}) \setminus \mathbf{E_{NM}}) \tag{2}$$

To increase the completeness of clusters, a further linkage step can refine the resulting clusters of $\mathbf{C_{rep}}$ by matching the clusters across each other. The comparison between clusters can be achieved by using cluster representatives such as one selected record of a cluster or a fused record representing the information of all records of a cluster. Due to the high complexity of the pairwise comparison, we do not consider missing merged clusters. This issue can be mitigated by employing alternative similarity computations or more relaxed classification thresholds to merge repaired clusters. For improved efficiency, cluster representatives can be used for comparisons instead of individual records.

Figure 1 shows the problem definition with three data sources $D1$ (triangle), $D2$ (circle), and $D3$ (pentagon) exemplary. Each of them consists of a set of records described by the attributes *description, name, brand*, and *id* indicating the same entity. The similarity graph SG consists of two connected components where the records representing entities 1 and 2 as well as 3 and 4 are in the same clusters. Due to the erroneous links (D1.1, D3.2'), (D1.1', D2.2), and (D2.4, D3.3), the first cluster consists of records representing entities 1 and 2. Additionally, the records D2.3 and D3.3 represent an entity with D2.4. The repair method aims to determine these incorrect edges so the records are split into 4 clusters $\mathbf{C_{rep}}$.

2.2 Graph Features

The core component for repairing the clusters involves using graph metrics to characterize edges. The current state-of-the-art approaches use **Graph neural networks (GNNs)**[6, 7, 19, 24, 49] to learn appropriate node or edge representations so-called embeddings. The derived embedding can be used for several downstream tasks such as node classification or link prediction. However, most approaches assume that the existing edges in the graph are correct so that information via the graph structure can be propagated using spectral- or spatial-based GNNs. However, the performance of GNNs highly depends on the amount of noise such as missing or wrong edges in a graph [16, 51, 52].

Instead of using GNNs to learn edge representations, we employ well-known centrality measures such as PageRank [5], closeness Centrality [42], and betweenness Centrality [17]. The intention of using centrality measures is based on the assumption that a record well-connected to other records in a cluster is likely to be correct. PageRank [5] was originally developed to determine the relevance of web pages based on their link structure. Closeness Centrality [42] measures how quickly a node can access other nodes in the network, with higher values indicating that a node is better positioned to reach others efficiently. Betweenness Centrality [17] quantifies the extent to which a node acts as a bridge along the shortest paths between other nodes, thereby highlighting its role in facilitating communication within the network.

In addition to the centrality of records, we incorporate characteristics related to ER, including similarity, strong link ratio, and normal link ratio introduced by Saeedi et al. [44]. These metrics are crucial, particularly in upholding the duplicate-free assumption for clean data sources. A strong link is an edge between two records where the similarity is the maximum of both adjacent nodes regarding a data source pair. Thus, the strong link ratio of a node is calculated by dividing the number of strong link edges by the total number of adjacent edges of a node.

These additional characteristics are instrumental in maintaining the integrity and accuracy of the data, ensuring that connections reflect true relationships within clean data sources.

3 Related Work

Data integration requires ER, a field extensively studied for decades [9, 12, 34]. The primary objective lies in identifying records that represent the same entity. Most of the approaches [14, 26, 32, 35] treat ER as a classification task, categorizing record pairs between two data sources as matches or non-matches. Recent research utilizes pre-trained [28, 29] and large language models [39] to address data sources with high heterogeneity.

However, due to quality issues and the transitivity of *equality* links, a pairwise view is insufficient to generate qualitative record clusters. The error-prone clusters can lead to wrong analysis results or incorrect downstream tasks such as knowledge graph construction. As a solution, methods for cluster repair become relevant to enhance the quality of derived clusters, building upon the determined.

In the following, we discuss cluster repair methods. Due to the requirement of training data for the classification model, we consider active learning methods. Additionally, we briefly discuss LLM-related ER since we use LLMs as oracles in our method.

Cluster repair: Hassanzadeh et al. [20] proposed an evaluation framework for single-pass clustering approaches for ER tasks without considering any constraints. CLIP [44] focuses on repairing clusters based on multiple duplicate-free data sources. The method categorizes the computed links into *strong*, *normal*, and *weak* links based on the similarity graph structure and the data sources. The different link categories distinguish if two records from two data sources are connected by an edge that maximizes the similarity for both records (*strong*), only one record

(*normal*), or none of them (*weak*). Using the link categories and the assumption of duplicate-free data sources, the method iteratively removes edges from clusters until they are *source-consistent*. The evaluation showed that CLIP outperforms the best-performing clustering methods from [20] considering duplicate-free data sources. The strong assumption regarding duplicate-free data sources is also utilized by [13, 48]. Lerm et al. [27] extended affinity propagation clustering applicable for clean and dirty data sources. In terms of the repair methods from Hassanzadeh et al. [20], the adapted cluster method outperformed the single pass-clustering algorithms. However, experiments showed that the approach leads to small clusters with high precision but at the cost of relatively low recall. To overcome the low recall, Saeedi et al. [43] proposed an agglomerative hierarchical clustering-based method using the basic strategies: *single-*, *complete-*, and *average*-linkage. To guarantee source consistency regarding clean data sources, they adapted the general method by adding certain constraints for merging clusters. Raad et al. [41] utilized the modularity as network information implicitly by applying the Louvain algorithm to determine incorrect links. Using the determined communities, the method computes an intra/inter-community link error degree for each edge. The authors evaluated the approach using Linked Open Data datasets to show its efficiency. However, the effectiveness was only shown by a small sample. Furthermore, selecting the error-degree threshold for the classification as non-match is a challenging task, as this is dataset-dependent and was only determined by sampling.

Active Learning: Due to the classification of links based on the graph metric-based features, training data is required. Active learning enables efficient and effective training data generation by determining informative unlabeled samples iteratively and interactively. In each iteration, the training data is utilized to determine new informative samples being labeled by an oracle. The algorithm terminates if a specific stop criterion is achieved such as the number of manually classified samples, the so-called labeling budget, or the performance of the current classification model is sufficient. Due to the lack of available training data, various methods [1, 11, 31, 37] have been proposed for ER. Mozafari et al. [31] proposed two approaches, named *Uncertainty* and *MinExpError*, being applicable for applications beyond ER. The main idea of these approaches is to use non-parametric bootstrapping to estimate the uncertainty of classifiers. Recent work [40] also addresses mutli-source ER problems introducing new challenges such as increasing heterogeneity and search space regarding informative samples.

LLM-based Entity Resolution: The application of large language models is established in various research fields including data management and wrangling tasks [15, 33]. Relevant to our approach are methods focusing on ER tasks [33, 39]. Peeters et al. [39] showed the feasibility of using LLMs for ER on various real-world datasets and different prompt variations without any fine-tuning. The results showed that the LLMs achieve comparable results or outperform existing methods based on pre-trained language models [29, 30]. However, the experiments only utilize small test sets so an end-to-end solution requires a high monetary effort. In terms of using LLMs as an oracle in the active learning step, Xiao et al. [50] used large language models in combination with fine-tuned language models to label informative samples in an active learning method and to filter out error-prone labeled data using the fine-tuned model. They evaluated their method considering NLP tasks.

In comparison to related work, we are the first who intensively utilize various network information as features for generating a classification model for cluster repair tasks. In contrast to the active learning method from Mozafari et al. [31], we focus on cluster repair tasks instead of ER problems. Therefore, we consider not only informative record pairs but also select informative pairs based on cluster-specific characteristics to determine representative training data regarding the various clusters. In contrast to the LLM-based ER methods, we do not only use the output of the model as the final decision rather we utilize the result for generating a classification model to

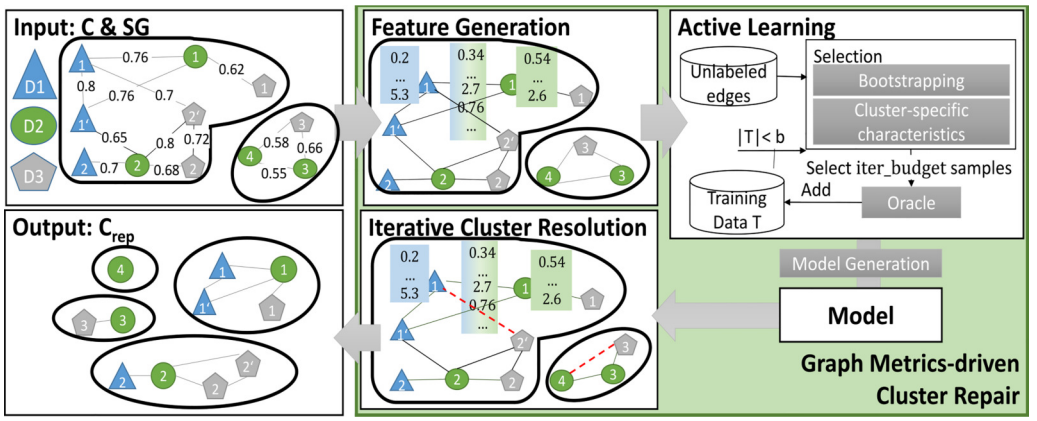


Fig. 2. Overview of the graph-based cluster repair method. The different shapes of the records represent the various data sources: D1: triangle, D2: circle, D3: pentagon. We highlighted the box, including the novel steps of our approach.

classify pairs as matches similar to the work of [50]. Moreover, the existing LLM-based methods do not focus on cluster repair methods rather than on pairwise linkage.

4 Graph-Based Cluster Repair

In the following, we will present our graph-based cluster repair method where the entire workflow is depicted in Figure 2. As input, the method has an initial set of clusters **C** derived from a similarity graph **SG** generated by an arbitrary record linkage framework. Given the dependency of the prediction power on the derived features and the similarities, we assume that the record linkage tool produces results where higher similarity scores between records are more likely to indicate matches. Furthermore, the generated similarity scores should be distributed across a broad range rather than being concentrated within a narrow interval. Similarly to CLIP [44], we also remove *weak links* being edges where both adjacent records are related to edges with a higher similarity.

Initially, we compute various graph metrics in the feature generation step, such as PageRank, and several centrality measures to construct node and edge features illustrated as boxes in the workflow Figure 2.

To determine the correctness of an edge, we classify each edge based on a trained model **M** using the derived edge features. The training process requires classified edges as matches and non-matches. Due to the sparsity of available training data, we execute an active learning method to select informative edges as training data **T**. In each iteration, we select $iter\_budget \geq 0$ unlabeled samples for labeling and add the labeled edge feature vectors to the existing training dataset **T**. The iteration terminates if the specified labeling budget b is exhausted. We extend the approach by considering cluster-specific characteristics such as the number of nodes. The intention is that the selected training data should represent the different available clusters. Therefore, the cluster-specific properties of the selected edges should be similar to the characteristics of all clusters. In the last step, we iteratively resolve the initial clusters **C** using the model **M** so that each group consists of records where the edges are classified as *match* with a high support. In the example of the workflow, the edges (D1.1, D3.2') and (D2.4, D3.3) are classified as non-match by the derived model using the features. The iterative cluster resolution step constructs repaired clusters starting with the records D1.1, D3.2', D2.4, and D3.3 being associated with edges classified as *non-matches*. In the iterations, the method extends the clusters adding the records to the cluster with the highest support.

4.1 Feature Generation

In our iterative cluster repair step (see Section 4.3), a classification model **M** classifies the edges in a cluster using graph metric-based feature vectors to determine if the edges are correct. Most approaches utilize the similarity of a link in combination with a clustering approach for cluster repair. The induced similarity graph of a cluster is utilized to derive various edge features in addition to the given similarity.

An overview of the considered graph-metrics is shown in Table 1. The main intuition of using graph metrics is to consider network information in addition to only local information such as similarity. To determine edge features based on node metrics, we compute the absolute value of the difference between the adjacent nodes. In addition to the common graph metrics, we also consider the link categories (see Section 3) by determining the *normal link ratio* and *strong link ratio*. The ratios determine how often a record is adjacent to a normal link or a strong link relative to the number of disjoint data sources being represented in the cluster. As an indicator of how well connected the cluster is, the *complete ratio* represents how similar the induced graph of a cluster is compared to a complete graph. Therefore, we compute the *complete ratio* by $|E|/\frac{|V|\cdot(|V|-1)}{2}$ where $|E|$ is the number of edges and $|V|$ the number of records.

Due to the same structure of different clusters, various edges can be represented by the same edge vector. Therefore, we only consider unique edge vectors in the training phase for generating the model **M**.

The efficiency of our approach is influenced by the complexity of the graph algorithms we use. Most of these algorithms have either cubic or quadratic complexity in relation to the number of vertices and edges in the graph. However, thanks to the blocking and classification steps, many edges are pruned, resulting in clusters with a moderate number of vertices and edges. As a result, the feature generation process requires significantly less runtime compared to the active learning model generation step shown in Section 5.

4.2 Cluster Characteristic Aware Active Learning with LLMs

Due to the high number of clusters and the lack of evaluated clusters, we need to generate training data efficiently. The current research provides various active learning methods for record linkage. In our case, we also want to consider cluster-specific characteristics to select edge feature vectors as training data. Therefore, we extend the active learning method of Mozafari et al. [31] by considering the number of nodes regarding the cluster of the selected edge. The main idea is to select iteratively a certain number $iter\_budget$ of unlabeled edge feature vectors being informative to extend the current training data **T**. The iterative process terminates if a stop criterion such as a total labeling budget b is reached. To determine informative vectors Mozafari et al. use a bootstrapping technique. The method generates k classifiers based on the current training dataset **T** by sampling with repetition. The determined models $m_1, .., m_k$ classify the unlabeled edge feature vectors where the predictions are utilized to compute the uncertainty $unc(\overrightarrow{e})$ of an edge e shown in Equation (3).

$$unc(\overrightarrow{e}) = \frac{\sum_{i=1}^{k} m_i(\overrightarrow{e})}{k} \cdot \left(1 - \frac{\sum_{i=1}^{k} m_i(\overrightarrow{e})}{k}\right) \tag{3}$$

The term $m_i(\overrightarrow{e})$ results in 0 or 1 if the edge e represents a non-match, respectively, a match.

We extend the uncertainty criterion by using the number of nodes regarding the graphs of each edge vector $\overrightarrow{e}$ as a cluster-specific characteristic. The goal is to avoid over and under-representing clusters with a certain size concerning all clusters. Note, that multiple clusters can contain edges with the same edge vector due to the same structure and similarity. Consequently, an edge vector

can be assigned to more than one cluster-specific characteristic. For selecting an edge vector $\overrightarrow{e}$, we determine a cluster-specific weight $w_c(\overrightarrow{e})$ based on the cluster size distribution from all available clusters d_C and the current training data distribution d_T. The distribution of d_C and d_T are represented as vectors with a dimension equal to the maximum number of nodes considering all clusters. An entry of d_C and d_T consists of the ratio between the frequency of clusters with a certain number of nodes and the total number of clusters in **C** resp. **T**. The weight according to a certain cluster size is computed by the difference $\overrightarrow{w} = d_C - d_T$. Due to the n:m relationship between edge feature vectors and clusters, we determine the average weight according to an edge feature vector $\overrightarrow{e}$ using the specific weights w_l of w where the lth entry corresponds to the cluster size of a cluster c where $e \in c$ holds.

In addition to the cluster-specific characteristic, we extend the selection strategy by using the average cosine distance between the unlabeled edge feature vector $\overrightarrow{e}$ and the already selected edge vectors $e_T \in T$. The intention is to choose dissimilar vectors compared to the current training dataset. Summarizing, the different measures $unc(\overrightarrow{e})$, $w(\overrightarrow{e})$ and $avg\_cos(\overrightarrow{e})$ are averaged to a final informativeness score being used to order the edge feature vectors. Using the order, we select $iter\_budget$ edge feature vectors in each iteration until the training data size achieves the labeling budget.

LLM oracle: Instead of using a human-based oracle, using large language models removes the manual effort of labeling the selected feature vectors. Consequently, the efficiency of generating labeled training data is enhanced, as its availability is no longer dependent on human involvement. The task of the LLM is to classify two records represented as JSON objects with the original given properties if they represent the same entity or not. Therefore, we define for each requested record pair the following prompt being sent using the LLM-specific API:
Classify record pairs represented as JSON objects as the same entity or not. Be aware that the content of a record can slightly differ due to quality issues such as typos or missing values. Output: yes/No
Entity A: a
Entity B: b

We process the response by searching for the keywords "yes" or "no." If the response contains any other content, we label the pair as a non-match by default. Due to the compact output, we can easily label the requested pairs and add them to the existing training data. The manually or automatically labeled edge vectors are utilized to train a classification model **M** for classifying edges as correct or not. In the following, we describe an iterative cluster repair method using the model **M**.

4.3 Iterative Cluster Resolution Step

The method resolves each cluster c utilizing the generated classification model **M**. The resolution step is shown in Algorithm 1. At first, we determine for each cluster c the set of edges E_{NM} and E_M classified as non-matches resp. as matches using the related edge feature vectors. Each record u and v of an edge $(u, v) \in E_{NM}$ represent a repaired cluster c_u and c_v of C_{rep} because u and v are classified as different entities. We iteratively merge the remaining records into the existing clusters until they are stable (lines 11–29). The process utilizes a support value $sup(r, c_{rep})$ indicating the strength of the assignment of record r and the cluster c_{rep}. The support is defined as the difference concerning the number of predicted matches and non-matches between the record r and records $t \in c_{rep}$ being already added to cluster c_{rep}. The merging process checks each cluster $c_{rep} \in C_{rep}$ by considering the adjacent nodes r of a record $s \in c_{rep}$ (line 17). The adjacent records r are determined by the adjacent edges E_s of the record s regarding the induced similarity graph of the cluster c returned by the function `graph()`(line 15). Suppose the adjacent record r is not connected via an edge predicted

ALGORITHM 1: Iterative Cluster Resolution Step

```
   Input:
   - M: classification model
   - c: cluster
   Output:
   - C_rep: set of repaired clusters
1  C_rep, E_NM, E_M ← ∅
2  for e = (u, v) ∈ edges(c) do
3      if classify (M, feature (e)) then
4          E_M ← E_M ∪ {e}
5      else
6          E_NM ← E_NM ∪ {e}
7          c_u = {u}
8          c_v = {v}
9          C_rep ← C_rep ∪ c_u ∪ c_v
10 change = true
11 while change do
12     change = false
13     for c_rep ∈ C_rep do
14         for s ∈ c_rep do
15             E_s ← adjacentEdges (graph(c), s)
16             processed ← ∅
17             for e_s = (r, s) ∈ E_s do
18                 if e_s ∉ E_NM ∧ r ∉ processed then
19                     c_r ← cluster (r, C_rep)
20                     if c_r = None then
21                         c_rep ← c_rep ∪ {r}
22                         change = true
23                     else
24                         sup_c_r =support (r, c_r, E_M, E_NM)
25                         sup_c_rep =support (r, c_rep, E_M, E_NM)
26                         if sup_c_rep > sup_c_r then
27                             c_r ← c_r \ {r}
28                             c_rep ← c_rep ∪ {r}
29                             change = true
30                 processed ← processed ∪ {r}
31 return C_rep
```

as non-match, is not processed in the current iteration (line 18), and is not merged into another cluster c_r (line 20). In that case, we add the adjacent record r to c_{rep}. Suppose the record r is already assigned to another cluster c_r. In that case, we determine and compare the support values sup_{c_rep} and sup_{c_r} using the induced graphs of c_{rep} and c_r as well as the set of classified non-matches E_{NM} and matches E_M (lines 24–26). We change the cluster assignment (lines 27/28), if the support sup_{c_rep} of the considered cluster c_{rep} is higher than the support sup_{c_r} of the previous cluster c_r for the record r. The assignment could differ depending on the order in which the adjacent records are processed. Therefore, we repeat this procedure for the new clusters until they do not change.

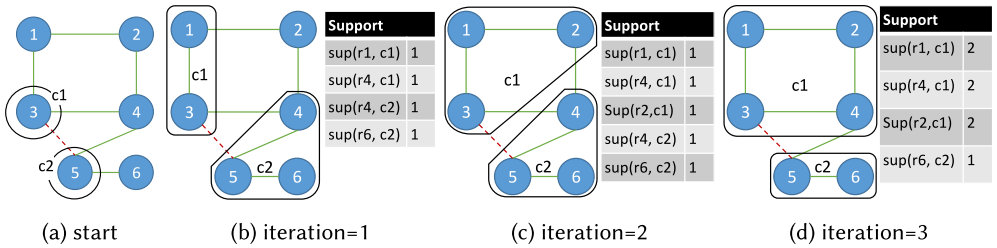


Fig. 3. Example of the iterative cluster repair procedure showing 6 records of an initial cluster. The dashed red line is an edge being classified as non-match.

Complexity: In the following, we briefly discuss the complexity of Algorithm 1 regarding a cluster c as input. The complexity depends on the number of edges E_c in the cluster c, the number of edges classified as matches $|E_M|$, and as non-matches $|E_{NM}|$. At the beginning, we partition each edge $e \in E_c$ into E_M and E_{NM} resulting in a complexity $O(|E_c|)$. The number of iterations in the while loop is bounded by the number of changes regarding the records to a cluster. The assignment can change due to higher support for a certain cluster compared to the support regarding a previously assigned cluster c'. The support depends on the number of positive edges E_M being associated with a record. Consequently, the while loop terminates in the worst case, if the algorithm has seen all edges in E_M. In each iteration of the while loop, the algorithm considers the adjacent records of each record of a cluster $c \in C_{rep}$ connected by an edge $e \in E_M$. Therefore, the total number of verified adjacent records is bounded by the number of edges in E_M. The complexity of the while loop is given by $O(|E_M| \cdot |E_M|)$ where the first factor represents the maximum number of iterations where the assignments can change and the second factor the total number of verified adjacent records (see lines 13–30). The complete complexity of Algorithm 1 is $O(|E_c| + |E_M|^2)$. regarding a certain cluster c.

Example: Figure 3 shows an example of the iterative cluster repair step. In this example, we have a cluster with 6 records. The classification model classifies the edge $(r3, r5) \in E_{NM}$ as non-match. Consequently, the records $r3$ and $r5$ represent two clusters $c1$ and $c2$. In the first iteration, the approach considers the records $r1$ being adjacent to $r3 \in c1$ as well as $r4$ and $r6$ connected to $r5 \in c2$. Note, that $r4$ will be considered for both clusters $c1$ and $c2$. Depending on the processing order of clusters, the records are merged into the clusters. Assuming $c2$ is the first cluster, $r4$ and $r6$ are added to it since both are adjacent to $r5$. After that, cluster $c1$ is considered where $r1$ and $r4$ are connected to $r3 \in c1$. Only $r1$ is assigned to the cluster $c1$ with a support of 1. The assignment of $r4$ does not change because the support $sup(r4, c1) = 1$ is equal to $sup(r4, c2) = 1$. In the next iteration of the while loop, the support of $r1, r4$, and $r6$ is the same. The record $r2$ is added to the cluster $c1$ with $sup(r2, c1) = 1$. Due to the added record $r2$ to cluster $c1$, the support $sup(r4, c1)$ changes to 2 in the third iteration. Therefore, the record $r4$ is assigned to $c1$ and removed from $c2$. In the next iteration, the assignment does not change so the approach terminates.

5 Evaluation

In this section, we evaluate our graph metric-based cluster repair method, beginning with the experimental setup describing the datasets, the default parameter configuration, and the considered metrics. We initially evaluate various graph metric combinations and compare them with the basic feature combination used in previous work [27, 43, 44] in Section 5.2. In Section 5.3, we analyze the impact of our modification to select informative links in the active learning step and the usage of LLMs as an oracle in Section 5.4. After that, we compare our method using a perfect oracle with

Table 2. Dataset Overview and Linking Configuration of MusicBrainz and Dexter

Dataset	Dexter			MusicBrainz	WDC
	C0	C50	C100		
#Records		21K		20K	5.3K
#Matches	368.5K	38K	16K	16.25K	6.8K

existing approaches in Section 5.5. Finally, we verify the robustness regarding erroneous similarities in Section 5.6.

5.1 Experimental Setup

5.1.1 Datasets. We use datasets from three domains: records about music albums (MusicBrainz), consumer products of type camera (Dexter), and computer hardware(subset of WDC). These datasets are multi-source datasets and heterogeneous regarding their error characteristics. In contrast to the MusicBrainz dataset being duplicate-free, the camera and WDC datasets are dirty and contain intra-source duplicates.

MusicBrainz: The MusicBrainz dataset is a synthetically generated dataset from the MusicBrainz (https://musicbrainz.org/) database. The dataset is corrupted by [22], consisting 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, length of values, and ratio of errors.

Dexter: This dataset is derived from the camera dataset of the ACM SIGMOD 2020 Programming Contest (http://www.inf.uniroma3.it/db/sigmod2020contest/index.html). The dataset consists of 23 sources with ≈21,000 records and intra-source duplicates. Each data source consists of source-specific attributes. In addition to the original dataset, Lerm et al. [27] also derive various datasets of different duplicate ratios. Therefore, they deduplicated a specified set of data sources and selected a certain ratio of records to construct clean data sources. The remaining data sources were used to generate the dirty data sources. We consider the datasets $C0, C50$, and $C100$ in our evaluation. For instance, the dataset $C50$ consists of ≈ 50% records from deduplicated sources according to the total number of records of the constructed dataset.

WDC: We utilize the WDC Training Corpus for Large-scale Product Matching, specifically a subset derived from the original dataset used in the Almser study . This subset includes data from four sources focused on hard drives. For our initial links, we employ the pre-computed similarity vectors available online.[1] Our analysis considers all records of the transitive closures derived from the true matches in the ground truth. By leveraging these transitive closures, we can identify all record pair combinations that are not present in the ground truth as matches.

We utilize the WDC Training Corpus for Large-scale Product Matching, specifically a subset derived from the original dataset used in the Almser study . This subset includes data from four sources focused on hard drives. For our initial links, we employ the pre-computed similarity vectors available online. Our analysis considers all records of the transitive closures derived from the true matches in the ground truth. By leveraging these transitive closures, we can identify all record pair combinations that are not present in the ground truth as matches.

5.1.2 Metrics&Parameters. We compute the F1-score, defined as the harmonic mean of precision and recall, to measure the effectiveness. We measure the efficiency utilizing the

[1] http://data.dws.informatik.uni-mannheim.de/benchmarkmatchingtasks/almser_gen_data/

Table 3. Evaluation Aspects

Characteristic	Comparison	Default parameters
Feature Generation (Section 5.2)	different graph metric feature combinations	perfect oracle, bootstrap AL
Active Learning (Section 5.3)	bootstrap vs extension (bootstrap ext)	perfect oracle, feature combination 0/1/2/3/4/5/6/7/8/9
LLM-based Oracle (Section 5.4)	different LLMs	feature combination 0/1/2/3/4/5/6/7/8/9, best AL for each dataset
Baseline Comparison (Section 5.5)	various cluster repair methods	perfect oracle, feature combination 0/1/2/3/4/5/6/7/8/9, best AL for each dataset
Robustness (Section 5.6)	different noise ratios	perfect oracle, feature combination 0/1/2/3/4/5/6/7/8/9, best AL for each dataset

runtime in seconds. Each experiment is repeated three times where we compute the Micro F1-score.

For the MusicBrainz and Dexter datasets, we utilized precomputed similarity graphs that were generated using various thresholds, as described in prior studies [27, 43–45]. These similarity graphs were computed using the FAMER framework [44] aggregating multiple similarity functions to produce similarity scores. Specifically, different thresholds were applied to classify the similarity scores and generate distinct similarity graphs. The thresholds were selected to maximize the F1-score of the resulting graphs. An overview of the datasets is provided in Table 2, while a detailed description of the linkage configurations can be found in Appendix A.

For the WDC, we utilize the provided feature vectors that were previously employed in research on Almser [40]. These vectors incorporate various string similarity measures for textual data and normalized absolute differences for numerical data. To construct a similarity graph, we train a random forest model using these computed similarity features. To enhance the training data selection, we apply the active learning procedure from Mozafari et al. [31], with a budget of 1000, $k = 50$, and $iter\_budget = 20$. For generating weights in the similarity graph SG, we use the prediction probabilities of matches. Therefore, we calibrate the random forest model using a 3-fold cross-validation. Since no threshold is applied, we represent it as -1 in the figures for clarity.

In pre-experiments, we determined the number of labels selected for each iteration $iter\_budget = 20$ and the number of models $k = 100$ for determining the uncertainty in the active learning step. As a classification model, we used the random forest implementation from `scikit-learn 1.1.2`. For computing the graph metrics we use `networkx 2.8.2`. As a default setting, we assume a perfect oracle in the active learning step. Table 3 shows an overview of the evaluation aspects regarding the compared components and the default parameters.

5.2 Graph Metric Analysis

In the first experiment, we evaluate various graph metric combinations to determine edge feature vectors for generating a classification model **M**. Overall, we test 8 combinations ranked regarding the F1-score and calculate the average rank over all datasets, all thresholds, and budgets. Figure 4 shows the F1-score for the top 2-ranked combinations, the largest combination, and the basic combination averaged over the labeling budget.

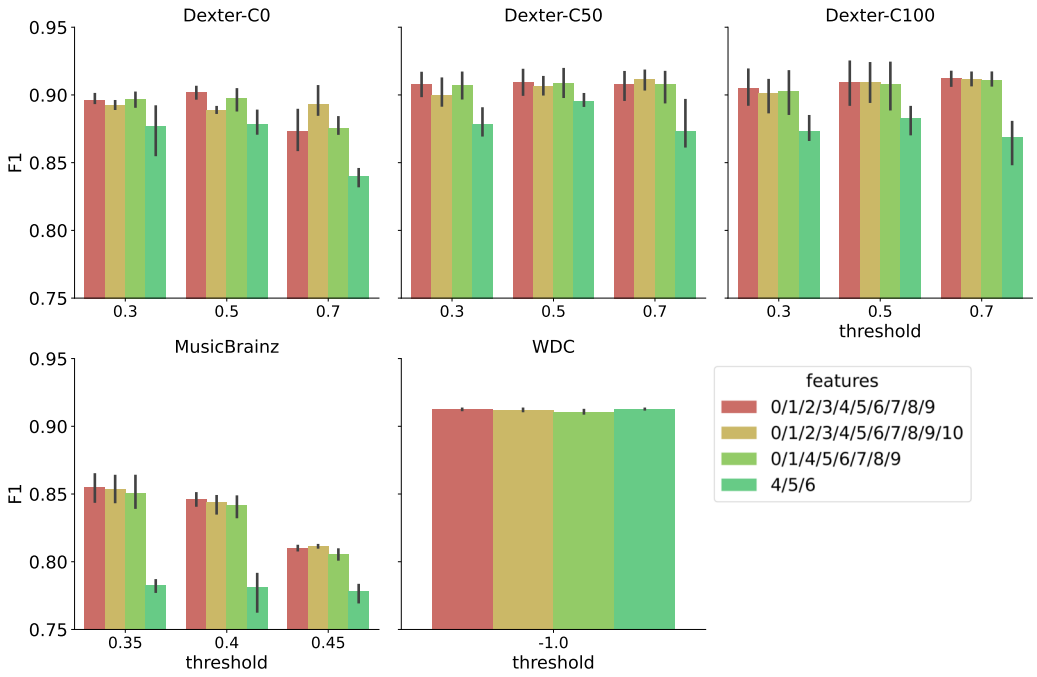
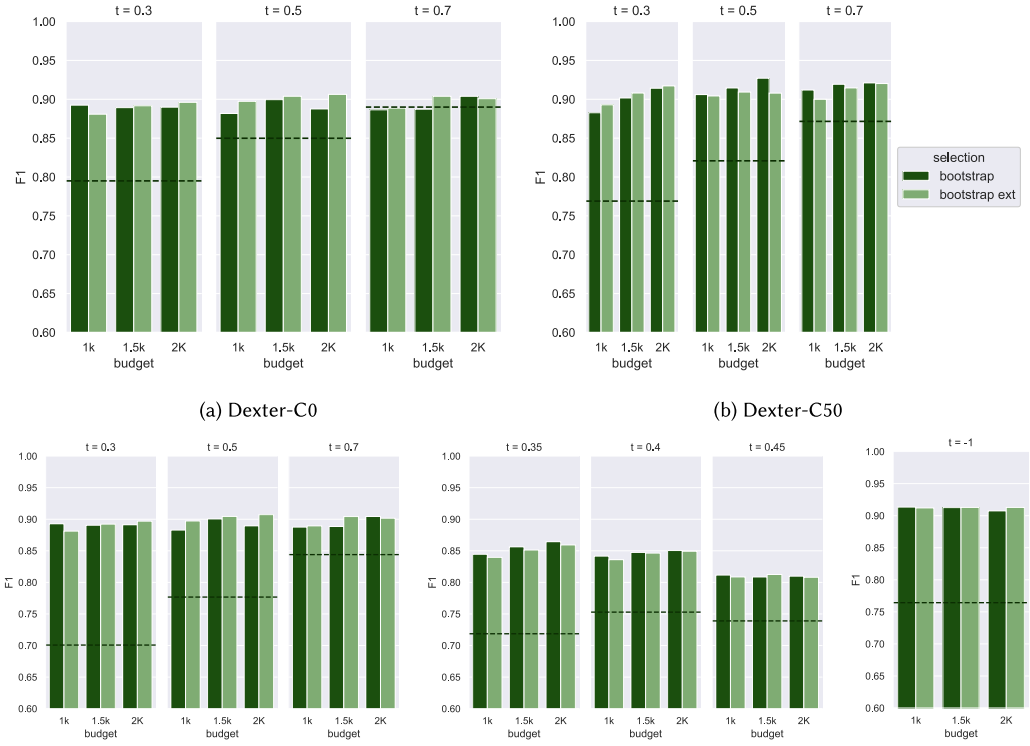


Fig. 4. Comparison of different graph metric combinations utilized as features to determine a classification model. 0:PageRank, 1:Closeness Centrality,2:Betweenness Centrality, 3:Cluster Coefficient, 4:Normal link ratio, 5:Strong link ratio, 6:Similarity, 7:Bridge, 8:Betweenness Centrality, 9:Complete ratio, 10:weighted degree.

We observe that the additional features improve the quality of resulting clusters compared to only using the similarity and link categories (features=4/5/6). The repair method generates similar results considering the top 2-ranked features and the largest one. Moreover, the combinations lead to stable results regarding different labeling budgets for the datasets C50 and C100 where we assume that the enriched features result in models with a high discriminative power even for small training datasets. For the WDC dataset, the different feature combinations achieve similar F1-score results. We explain that due to the small clusters and the small number of edges per cluster. Therefore, the additional network information does not lead to a significant improvement compared to the smallest feature combination. Moreover, we assume that the determined prediction probabilities are more accurate than the aggregated similarities. Therefore, the similarity and the link category features are sufficient in this case.

5.3 Cluster-Specific Training Data Selection

We compare the cluster-specific active learning method (bootstrap ext) with the original approach by Mozafari et al. [31] (bootstrap) shown in Figure 5. Both selection strategies improve the initial quality considering the given similarity graph (dashed line) for both datasets and similarity thresholds using a labeling budget above 1500. The results also show robustness using different labeling budgets since the F1-score only differs by ≈0.017 (bootstrap) and ≈0.016 (bootstrap ext). However, we observe a negative effect for the dirty dataset C0 and high thresholds such as 0.7. The approach cannot improve the linkage results for a small labeling budget. We assume that the filtering effect using a high threshold impedes the selection of informative pairs.



(a) Dexter-C0 (b) Dexter-C50

(c) Dexter-C100 (d) MusicBrainz (e) WDC

Fig. 5. Results considering the basic selection strategy (bootstrap) and the cluster-specific selection (bootstrap ext) in the active learning step. The green line represents the quality of the input similarity graph.

In terms of the effectiveness regarding data quality issues, the cluster-specific selection strategy slightly improves the results by up to ≈0.018 compared to the original active learning method for repairing clusters of dirty datasets such as C0. However, the baseline achieves better results for duplicate-free datasets such as MusicBrainz and C100 than the extended selection. Concerning the WDC dataset, both active learning procedures result in similar quality. We assume that the clusters show similar sizes so that the difference between both selection strategies is marginal. We conclude that the cluster characteristics are more relevant for selecting samples where the clusters are larger due to more links based on the intra-source record comparisons. In contrast to the dirty data sources, we observe that the baseline performs better for clean data sources indicating that the cluster-specific characteristics are less relevant in these scenarios and negatively impact the informativeness score of an edge. The labeling budget has a marginal positive influence on the results, indicating that our method is capable of generating high-quality clusters even with a limited budget.

5.4 LLMs for Active Learning

For comparison, the usage of LLMs as oracles in the active learning step, we initially considered a labeling budget of 500 and the Dexter C0 dataset to select an appropriate model. We compare our method using the perfect oracle and the following LLM models: `gpt-3.5-turbo`, `gpt4`, `claude-3-opus` with the default configuration. Table 4 shows the F1-score results and the

Table 4. F1-Score Results of the Active Learning Step Using a Perfect Oracle Compared with Different Large Language Models `gpt-3.5-turbo`, `gpt-4`, and `claude-3-opus` as an Oracle on Dexter-C0 with a Labeling Budget of 500 as well as the Quality of the Input Similarity Graph

t	Input	Perfect	gpt-3.5-turbo	gpt-4	claude-3-opus
0.3	0.8	**0.88**	0.82 (0.65)	0.86 (0.72)	0.87 (0.68)
0.5	0.85	**0.88**	0.88(0.65)	0.83 (0.67)	0.82 (0.64)
0.7	**0.89**	0.835	0.82(0.74)	0.83 (0.87)	0.83 (0.87)

The accuracy of the oracle is shown in brackets.

accuracy of the requested record pairs on the Dexter dataset C0 for a labeling budget of 500. For a heterogeneous set of record pairs existing in impure similarity graphs, the LLMs-based approaches can generate models that improve the input similarity graph, although the accuracy of the LLM is at most 0.67 for thresholds of 0.3 and 0.5. Similar to the previous analysis, the presented approach has problems with clean similarity graphs generated by higher thresholds and a small labeling budget, as the distinction between matches and non-matches based on a small training dataset and only small differences is very challenging. Therefore, the model with the perfect ground truth also decreases the quality of the linkage result and the LLM-based methods do not lead to any improvement although the accuracy is above 0.8. To summarize, using the various LLM models as an oracle can improve the initial clusters. However, due to the low accuracy of the labels, our cluster repair model achieves smaller F1 scores than a perfect oracle. Therefore, methods for denoising class labels [18, 38] are promising directions for future work. All models perform similarly with regard to the achieved quality. Interestingly, although the accuracy of `gpt-3.5-turbo` is lower than that of `gpt-4`, the F1-score is comparable or higher (t=0.5). We explain that `gpt-3.5-turbo` labels the more relevant pairs regarding the model correctly than the `gpt-4` model.

In the following, we will utilize the `gpt-3.5-turbo` model concerning a labeling budget of 1000, 1500, and 2000. We test the large language model-based method on all datasets. We generate for each dataset a fine-tuned model using the first 20 record pairs by our active learning approach concerning the threshold 0.35 (MusicBrainz), 0.3 (Dexter-C0/C50/C100), and the record pairs of the WDC dataset. The comparison between the perfect oracle and the LLM-based oracle without and with fine tuning is depicted in Figure 6. The fine-tuned method achieves comparable results according to the perfect oracle for the homogeneous dataset MusicBrainz, and the small WDC dataset. In contrast to the MusicBrainz and WDC dataset, the results of the Dexter datasets show the challenges of using LLMs as an oracle. Due to the high heterogeneity, the F1-score differs from 0.021 to 0.078 for the basic LLM model and from 0.011 up to 0.063 for C0. We also observe the effect that the quality does not increase for larger labeling budgets, which is what we would expect. We explain that due to the low accuracy between 0.56 and 0.73 of the LLM-based oracles. Moreover, we assume that the noisy training data influences the selection of informative pairs negatively so that the determined training data consists of noisy labeled data and potentially non-informative pairs for larger budgets. In terms of the Dexter datasets, we also observe that only the fine-tuned models for the threshold 0.3 can achieve better results than the base models. We explain this due to the high resulting heterogeneity within the similarity graphs so that in this case fine-tuning should be performed for each similarity graph generated by a certain threshold separately. Summarizing, the LLM-based oracle leads to comparable results for certain cases. The main issue of the determined labels is the small accuracy. Therefore, a promising solution for future work might be the application of denoising methods such as [38].

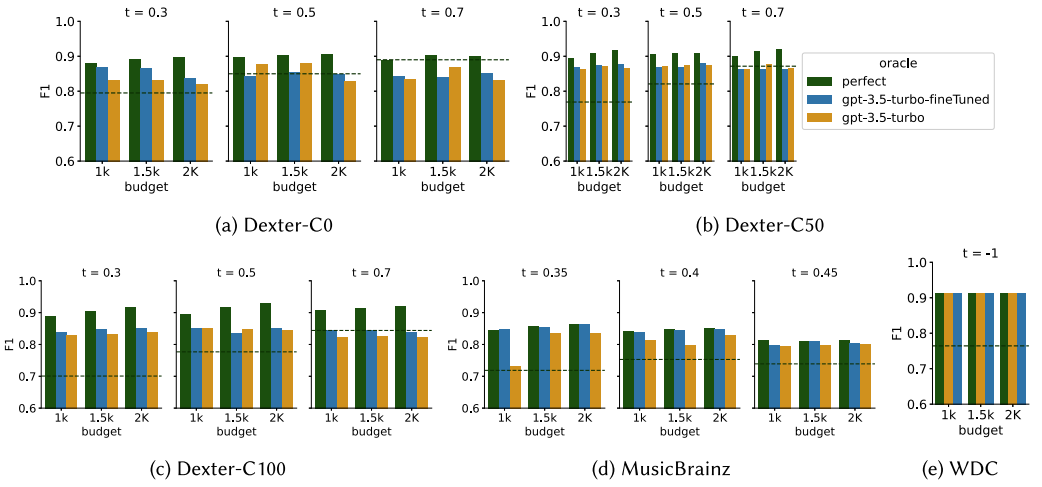


Fig. 6. Comparison between the perfect oracle, `gpt-3.5-turbo` and `gpt-3.5-turbo-fineTuned` with 20 samples labeled by the ground truth.

5.5 Comparison with Existing Repair Methods

We compare our method using a perfect oracle with the CLIP method [44], two clustering-based repair approaches based on hierarchical agglomerative clustering with the different clustering strategies *single* (MSCD S-Link), *complete* (MSCD C-Link), and *average* (MSCD A-Link) [43], affinity propagation (MSCD-AP) [27] as well as the Louvain-based error degree method (err-louvain) [41] described in Section 3. The original method [41] considers unweighted graphs. To guarantee a fair comparison, we also evaluate the method utilizing the similarities named err-louvain-sim.

Parameters: CLIP is a parameter-free approach since it is based on a greedy algorithm. The affinity propagation (MSCD-AP) [27] consists of parameters to guarantee convergence. The authors integrated a parameter adaptation step that is applied if the method does not find a solution. The agglomerative clustering approaches [43] use a similarity threshold to control the merging process of clusters. The authors set the merging threshold in their evaluation to the minimum threshold to generate the input similarity graph. Detecting error-prone links requires the specification of a $err\_threshold$ where an edge is classified as incorrect if the intra or inter-error degree is above the specified threshold. We evaluated various error thresholds between 0.1 and 0.5 where $err\_threshold = 0.2$ and $err\_threshold = 0.3$ for err-louvain resp. err-louvain-sim achieved qualitative results in pre-experiments overall.

Depending on the datasets' dirtiness and the clustering algorithm's configuration, the existing methods improve the quality of the initial similarity graphs as shown in Figure 7. However, the resulting quality of a certain method highly differs regarding the different duplicate ratios and thresholds considering the Dexter dataset. For instance, the F1-score achieved by CLIP ranges from 0.1(C0) to 0.9(C100). The hierarchical clustering methods (MSCD S/C/A-Link) achieve comparable results for the WDC dataset. However, the results of the hierarchical clustering methods designed for mixed datasets differ between 0.81(C0) and 0.9(C100) for MSCD A-LINK concerning all datasets. Similarly, the graph metric-based method utilizing the Louvain algorithm results in a moderate variation from 0.83(C100) to 0.89(C0) using the record similarity for the Dexter dataset. However, for the MusicBrainz and the WDC dataset, the err-lovain-sim method performs poorly achieving at most an F1-score of 0.55 (MusicBrainz) and below 0.18 for WDC. In contrast, the err-louvain method without using the similarities performs poorly for the Dexter dataset where the F1-score

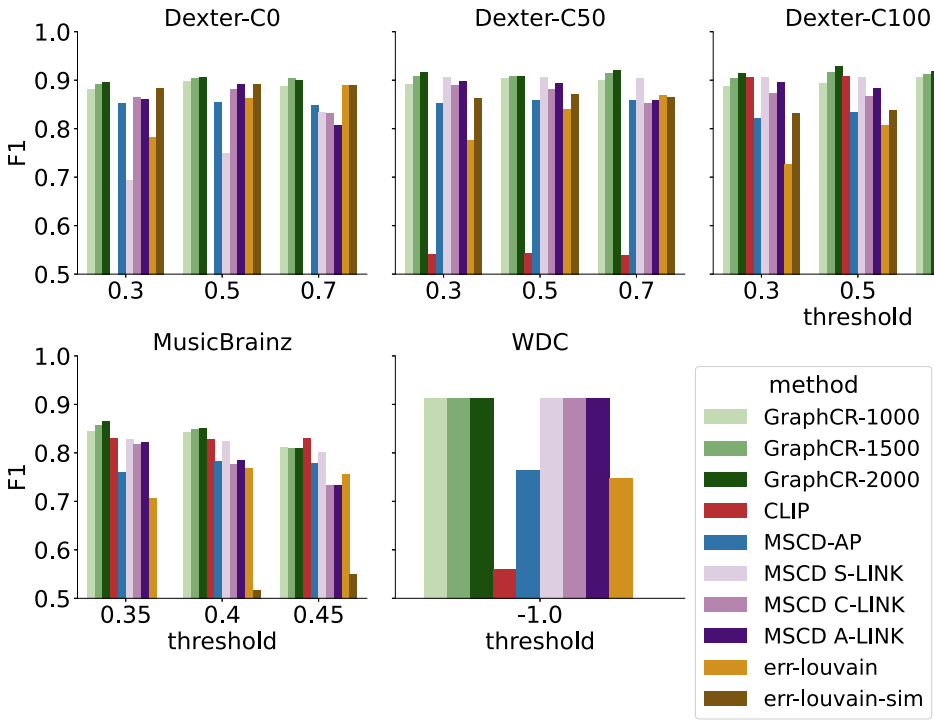


Fig. 7. F1-score results of our proposed approach (GraphCR) as compared with the other repair methods CLIP [44], affinity propagation clustering (MSCD-AP) [27], agglomerative hierarchical clustering methods [43] with the different variations regarding the merging step single (MSCD S-LINK), complete (MSCD C-LINK) and average (MSCD A-LINK) as well as the louvain-based method [41] with (err-louvain-sim) and without (err-louvain) the usage of similarities.

differs from 0.73(C100) to 0.89(C0). GraphCR outperforms the existing approaches concerning the used thresholds and datasets for labeling budgets ≥ 1500 by up to 0.8(CLIP), 0.17(err-louvain), 0.41(err-louvain-sim), 0.1(MSCD-AP/MSCD A-Link), 0.09(MSCD C-Link), and 0.2(MSCD S-link). The F1-score differences regarding the dirtiness of datasets are tiny compared to the SOTA approaches. For instance, the F1-score difference is smaller than 0.03 for GraphCR concerning C0 and C100 using a labeling budget of 1500 overall thresholds.

To properly compare the performance of these approaches, we rely on the Bayesian analysis proposed in [2]. To determine significant differences between the two approaches, we rely on a Bayesian signed rank test [3]. Using Bayesian statistics gives the advantage over a frequentist hypothesis testing approach that can not only reject but also accept a null hypothesis. We can also define a *region of practical equivalence* (ROPE) where approaches perform equally well. The `Autorank` [21] package provides these methods and can automatically set the ROPE in relation to effect size. The result of our analysis then gives us a probability that one approach is better (or worse) than another. We provide a verdict if one of these probabilities is $\geq 95\%$, or else we see the result as inconclusive.

The result of this analysis is a decision matrix in Figure 8. We compared the F1-score of each approach across all dataset/threshold combinations. For GraphCR, we also compare different labeling budgets. It is evident that GraphCR, with a budget of 2000, is significantly better than all other approaches. Even with a budget of 1500, it is better than all approaches that are not GraphCR.

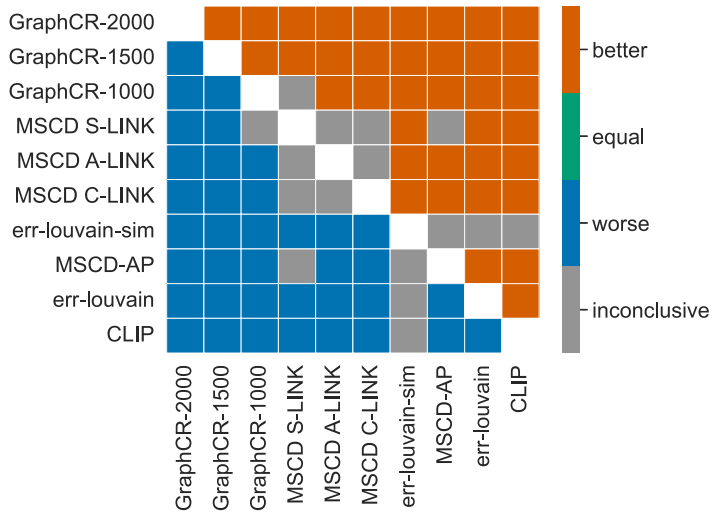


Fig. 8. Decision matrix comparing cluster repair approaches using Bayesian signed rank tests. Each cell shows the decision when comparing the row approach to the column approach.

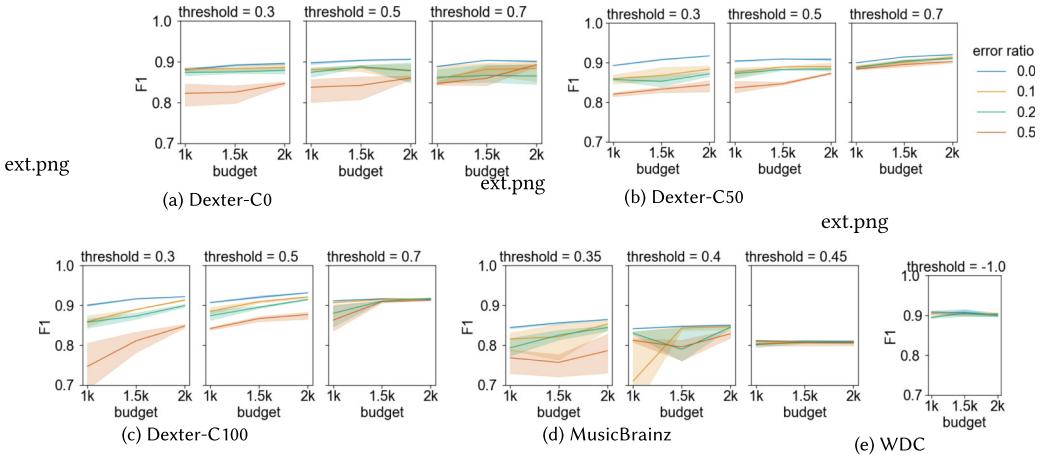


Fig. 9. Results with various error ratios of the similarities using the basic and extended selection strategy.

5.6 Effect of Noisy Similarities

Additionally, of various threshold settings resulting in different similarity graphs, we also evaluate the impact of erroneous similarities. Reasons for inappropriate similarities can be caused by inappropriate similarity functions or improperly configured ER systems. Within this section, we aim to assess the robustness of our proposed method in the presence of noisy similarities. In the following experiment, we randomly select a specified ratio of edges and set the similarity to a random number between 0 and 1. We assume that the noisy similarity edges negatively impact our classification model and, therefore, the result of the cluster repair.

Figure 9 shows the results for the MusicBrainz dataset applying the bootstrapping selection (bootstrap) and the Dexter dataset C0 using the extended strategy (bootstrap ext). Using the original similarity graph without any noise (error ratio = 0) for the MusicBrainz dataset results in a small

Table 5. Runtime Results (s) of the Complete Workflow t_{total}, the Feature Generation Step t_{feat} as well as the Resolving Algorithm t_{res} Considering the Different Datasets for the Similarity Graphs Based on Various Thresholds

dataset	threshold	$\|C\|$	$\overline{\|V\|}$	$\overline{\|E\|}$	t_{feat}	t_{res}	t_{total}
C0	0.30	1729	11.41	180.71	79.13	2.16	572.23
C0	0.50	1850	10.64	162.06	79.86	2.00	558.23
C0	0.70	2110	9.15	132.38	67.57	1.62	479.36
C50	0.30	994	7.39	43.62	4.23	0.44	138.95
C50	0.50	1046	6.93	38.12	3.88	0.56	127.50
C50	0.70	1181	5.86	27.83	3.16	0.70	96.18
C100	0.30	962	6.03	21.44	3.37	0.70	151.16
C100	0.50	1004	5.67	18.15	3.30	0.67	148.30
C100	0.70	1119	4.77	12.59	2.21	0.80	111.25
MB	0.35	3657	3.78	4.51	11.10	1.56	74.34
MB	0.40	3984	3.20	3.54	5.81	2.13	88.95
MB	0.45	4022	2.90	2.99	2.89	1.19	48.30
WDC	-1	532	2.9	2.42	0.085	0.2	17.96

We use a labeling budget of 2000. We also present the numbers of clusters $|C|$, average numbers of vertices and edges per cluster $\overline{|V|}$ resp. $\overline{|E|}$.

quality difference of at most 0.007. As expected, the increase in the error ratio leads to a decrease in quality. Especially, an error ratio of 0.5 leads to F1 score differences up to 10%(MusicBrainz) and 6%(Dexter-C0). The threshold and the budget influence the robustness of the results. An increasing threshold and labeling budget positively influence the robustness. A higher threshold reduces the number of noisy similarities already included in the original graph compared to a smaller threshold. Moreover, the increasing budget leads to a reduced impact of noisy links. However, the increasing budget does not lead to robust results for the MusicBrainz dataset regarding a high error ratio of 0.5 and a small threshold of 0.35. The results of the WDC dataset show that qualitative similarities lead to robust results even for a high error ratio. Summarizing, we observe that outliers of similarity highly influence the quality of the result from the cluster repair process. The influence is larger than the impact of the threshold where the results are robust as shown in previous experiments. Therefore, selecting appropriate similarity functions or pre-trained language models is essential to compute meaningful similarities. When comparing the effects of the threshold and the labeling budget, it is evident that the threshold has a greater influence due to its role in filtering out erroneous similarities within the original graph.

5.7 Runtime Analysis

To analyze our approach's effectiveness, we consider the runtimes using the different datasets with regard to various similarity graphs depending on the threshold. We ran our experiments on an 8-core AMD Ryzen 7 Pro 7840U 3.3 GHz CPU and 32GB of RAM. Table 5 shows statistics about the initial clusters of C, the total runtime as well as the runtimes for the feature generation and the resolution step (see Algorithm 1). As expected, the runtime scales with the size of the initial similarity graph, the higher the number of edges is the higher the runtime is. The main components of the runtime are the active learning and the model generation step. Especially, the active learning step scales with the number of edges, the label budget, and the number of models k to determine

the *uncertainty* (see Equation (3)). The cluster resolution algorithm has only a small impact on the total runtime.

In our analysis, we observed that our method exhibits slower performance than baseline unsupervised algorithms. This is primarily due to the inclusion of an additional active learning and model generation step in our approach. The runtimes for the unsupervised methods across the various datasets are as follows: CLIP completes its processing in under 50 seconds, MSCD-AP takes less than 295 seconds, and MSCD-S/C/A operates in under 5 seconds. The err-louvain algorithm achieves a runtime of less than 1.3 seconds.

5.8 Summary

Our results demonstrate that the graph-based cluster repair approach significantly enhances the quality of initial clusters with a moderate labeling effort. This method surpasses existing cluster repair techniques while requiring less configuration effort than traditional approaches. A key factor in our success is the extensive utilization of graph metric features, which provide superior performance compared to methods relying solely on similarity and link category-based features, as detailed in Section 5.2. Furthermore, our extended active learning method improves efficacy on noisy datasets compared to the standard bootstrap-based selection strategy. When employing LLM-based oracles, we observe comparable results, contingent upon the fine-tuning of the LLM model. However, it is important to note that our method yields moderate performance when the similarities within the data are predominantly noise.

6 Conclusion

Cluster repair methods play an integral role in multi-source ER tasks being relevant for data integration. The main goal of ER is identifying records representing the same entity. The majority of ER systems focus on pairwise linkage. However, due to the transitive closure of equality links and incorrect edges, the result potentially consists of clusters with records representing different entities. Therefore, cluster repair methods are required to identify incorrectly assigned records utilizing the generated similarity graph. The majority of repair methods intensively rely on the assumption of duplicate-free data sources. Current methods try to overcome this issue by modifying general clustering approaches. However, the results vary depending on the configuration of the method and the degree of dirtiness.

Therefore, we proposed a novel cluster repair method relying on graph metric features, enabling a classification model's training. The computed model is used to modify the initial clusters by iteratively removing edges being classified as non-match. Due to the lack of available training data, we integrate and extend an active learning method by considering cluster-specific characteristics to guarantee that the selected training samples represent the complete dataset. Moreover, we integrate the usage of large language models as an oracle to decrease the manual effort. We evaluated our proposed approach using two datasets. The results showed that our method outperforms existing approaches using a moderate labeling budget. We also showed the feasibility of using large language models as oracles.

In future work, we plan to consider graph-structured data sources such as knowledge graphs and utilize the semantic graph structure in combination with the similarity graph-based metrics. Due to the manual labeling process in the active learning step, we will evaluate graph-based methods [40] to reduce the number of selected samples. Moreover, we will also consider cluster-wise active learning strategies where a whole cluster is labeled regarding the correct and incorrect record. This allows the application of graph augmentation, such as the addition of correct links regarding one cluster. In the direction of using LLMs as oracles, we will consider combination strategies of multiple models and look at methods for denoising the predictions of LLMs.

Availability

Reference code and datasets are available from our repository at https://github.com/vicolinho/graphCR and https://www.dropbox.com/scl/fo/xtawtr2qbeckt0esu5v0h/h?rlkey=6lkz7hwsyhnqgyhg9quh7pp96&dl=0.

References

[1] Kedar Bellare, Suresh Iyengar, Aditya G. Parameswaran, and Vibhor Rastogi. 2012. Active sampling for entity matching. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1131–1139. DOI: https://doi.org/10.1145/2339530.2339707

[2] Alessio Benavoli, Giorgio Corani, Janez Demšar, and Marco Zaffalon. 2017. Time for a change: A tutorial for comparing multiple classifiers through bayesian analysis. *Journal of Machine Learning Research* 18, 77 (2017), 1–36. Retrieved from http://jmlr.org/papers/v18/16-305.html

[3] Alessio Benavoli, Giorgio Corani, Francesca Mangili, Marco Zaffalon, and Fabrizio Ruggeri. 2014. A Bayesian Wilcoxon signed-rank test based on the Dirichlet process. In *Proceedings of the 31st International Conference on Machine Learning, ICML (JMLR Workshop and Conference Proceedings, Vol. 32)*. JMLR.org, 1026–1034. Retrieved from http://proceedings.mlr.press/v32/benavoli14.html

[4] Jens Bleiholder and Felix Naumann. 2008. Data fusion. *ACM Comput. Surv.* 41, 1 (2008), 1:1–1:41. DOI: https://doi.org/10.1145/1456650.1456651

[5] Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. *Comput. Networks* 30, 1–7 (1998), 107–117. DOI: https://doi.org/10.1016/S0169-7552(98)00110-X

[6] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral networks and locally connected networks on graphs. In *Proceedings of the 2nd International Conference on Learning Representations, ICLR 2014*. Retrieved from http://arxiv.org/abs/1312.6203

[7] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. 2019. Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019*. ACM, 257–266. DOI: https://doi.org/10.1145/3292500.3330925

[8] Peter Christen. 2008. Febrl: An open source data cleaning, deduplication and record linkage system with a graphical user interface. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1065–1068.

[9] Peter Christen. 2012. *Data Matching – Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Springer. DOI: https://doi.org/10.1007/978-3-642-31164-2

[10] Peter Christen. 2012. A survey of indexing techniques for scalable record linkage and deduplication. *IEEE Trans. Knowl. Data Eng.* 24, 9 (2012), 1537–1555.

[11] Victor Christen, Peter Christen, and Erhard Rahm. 2019. Informativeness-based active learning for entity resolution. In *Machine Learning and Knowledge Discovery in Databases - Proceedings of the International Workshops of ECML PKDD (Communications in Computer and Information Science, Vol. 1168)*. Springer, 125–141. DOI: https://doi.org/10.1007/978-3-030-43887-6_11

[12] 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: https://doi.org/10.1145/3418896

[13] Gerard de Melo. 2013. Not quite the same: Identity constraints for the web of linked data. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence*. AAAI Press, 1092–1098. DOI: https://doi.org/10.1609/AAAI.V27I1.8468

[14] 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: https://doi.org/10.1145/3405476

[15] Raul Castro Fernandez, Aaron J. Elmore, Michael J. Franklin, Sanjay Krishnan, and Chenhao Tan. 2023. How large language models will disrupt data management. *Proc. VLDB Endow.* 16, 11 (2023), 3302–3309. DOI: https://doi.org/10.14778/3611479.3611527

[16] James Fox and Sivasankaran Rajamanickam. 2019. How robust are graph neural networks to structural noise? arXiv:1912.10206. Retrieved from http://arxiv.org/abs/1912.10206

[17] Linton C. Freeman. 1977. A set of measures of centrality based on betweenness. *Sociometry* 40, 1 (1977), 35–41. Retrieved June 12, 2025 from http://www.jstor.org/stable/3033543

[18] Shivani Gupta and Atul Gupta. 2019. Dealing with noise problem in machine learning data-sets: A systematic review. *Procedia Computer Science* 161 (2019), 466–474. DOI: https://doi.org/10.1016/j.procs.2019.11.146

[19] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems 30: Proceedings of the Annual Conference on Neural Information Processing Systems 2017*. 1024–1034.

[20] Oktie Hassanzadeh, Fei Chiang, Renée J. Miller, and Hyun Chul Lee. 2009. Framework for evaluating clustering algorithms in duplicate detection. *Proc. VLDB Endow.* 2, 1 (2009), 1282–1293. DOI: https://doi.org/10.14778/1687627.1687771

[21] Steffen Herbold. 2020. Autorank: A python package for automated ranking of classifiers. *J. Open Source Softw.* 5, 48 (2020), 2173. DOI: https://doi.org/10.21105/JOSS.02173

[22] 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: https://doi.org/10.1109/TBDATA.2016.2637378

[23] P. W. Holland and S. Leinhardt. 1971. Transitivity in structural models of small groups. *Comp. Group Stud.* 2, 2 (1971), 107–124. DOI: https://doi.org/10.1177/104649647100200201

[24] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.

[25] Hanna Köpcke and Erhard Rahm. 2008. Training selection for tuning entity matching.. In *Proceedings of the QDB/MUD*. Auckland, 3–12.

[26] Hannah Köpcke, Andreas Thor, and Erhard Rahm. 2010. Learning-based approaches for matching web data entities. *IEEE Internet Comput.* 14, 4 (2010), 23–31. DOI: https://doi.org/10.1109/MIC.2010.58

[27] 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)*. 217–236. DOI: https://doi.org/10.18420/btw2021-11

[28] Bing Li, Yukai Miao, Yaoshu Wang, Yifang Sun, and Wei Wang. 2021. Improving the efficiency and effectiveness for BERT-based entity resolution. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*. AAAI Press, 13226–13233. DOI: https://doi.org/10.1609/AAAI.V35I15.17562

[29] 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: https://doi.org/10.14778/3421424.3421431

[30] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. Retrieved from https://arxiv.org/abs/1907.11692

[31] 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 Endow.* 8, 2 (Oct. 2014), 125–136.

[32] 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: https://doi.org/10.1145/3183713.3196926

[33] Avanika Narayan, Ines Chami, Laurel J. Orr, and Christopher Ré. 2022. Can foundation models wrangle your data? *Proc. VLDB Endow.* 16, 4 (2022), 738–746. DOI: https://doi.org/10.14778/3574245.3574258

[34] Markus Nentwig, Michael Hartung, Axel-Cyrille Ngonga Ngomo, and Erhard Rahm. 2017. A survey of current link discovery frameworks. *Semantic Web* 8, 3 (2017), 419–436. DOI: https://doi.org/10.3233/SW-150210

[35] Axel-Cyrille Ngonga Ngomo, Mohamed Ahmed Sherif, Kleanthi Georgala, Mofeed Mohamed Hassan, Kevin Dreßler, Klaus Lyko, Daniel Obraczka, and Tommaso Soru. 2021. LIMES: A framework for link discovery on the semantic web. *Künstliche Intell.* 35, 3 (2021), 413–423. DOI: https://doi.org/10.1007/S13218-021-00713-X

[36] Axel-Cyrille Ngonga Ngomo, Mohamed Ahmed Sherif, and Klaus Lyko. 2014. Unsupervised link discovery through knowledge base repair. In *The Semantic Web: Trends and Challenges - Proceedings of the 11th International Conference, ESWC 2014, Proceedings (Lecture Notes in Computer Science, Vol. 8465)*. Springer, 380–394. DOI: https://doi.org/10.1007/978-3-319-07443-6_26

[37] Axel-Cyrille Ngonga Ngomo and Klaus Lyko. 2012. EAGLE: Efficient active learning of link specifications using genetic programming. In *Proceedings of the International Conference on The Semantic Web: Research and Applications*. Berlin, 149–163.

[38] Curtis Northcutt, Lu Jiang, and Isaac Chuang. 2021. Confident Learning: Estimating uncertainty in dataset labels. *J. Artif. Int. Res.* 70 (May 2021), 1373–1411. DOI: https://doi.org/10.1613/jair.1.12125

[39] Ralph Peeters and Christian Bizer. 2023. Using ChatGPT for entity matching. In *New Trends in Database and Information Systems - ADBIS 2023 Short Papers, Doctoral Consortium and Workshops: AIDMA, DOING, K-Gals, MADEISD, PeRS, Barcelona, Spain, September 4-7, 2023, Proceedings (Communications in Computer and Information Science)*, Alberto Abelló, Panos Vassiliadis, Oscar Romero, Robert Wrembel Francesca Bugiotti, Johann Gamper, Genoveva Vargas-Solar, and Ester Zumpano (Eds.). Springer, 221–230. DOI: https://doi.org/10.1007/978-3-031-42941-5_20

[40] Anna Primpeli and Christian Bizer. 2021. Graph-boosted active learning for multi-source entity resolution. In *The Semantic Web - ISWC 2021 - Proceedings of the 20th International Semantic Web Conference, ISWC 2021 (Lecture Notes in Computer Science, Vol. 12922)*. Springer, 182–199. DOI: https://doi.org/10.1007/978-3-030-88361-4_11

[41] Joe Raad, Wouter Beek, Frank van Harmelen, Nathalie Pernelle, and Fatiha Saïs. 2018. Detecting erroneous identity links on the web using network metrics. In *The Semantic Web - ISWC 2018 - Proceedings of the 17th International Semantic Web Conference, Part I (Lecture Notes in Computer Science, Vol. 11136)*. Springer, 391–407. DOI: https://doi.org/10.1007/978-3-030-00671-6_23

[42] Gert Sabidussi. 1966. The centrality index of a graph. *Psychometrika* 31, 4 (Dec 1966), 581–603. DOI: https://doi.org/10.1007/BF02289527

[43] Alieh Saeedi, Lucie David, and Erhard Rahm. 2021. Matching entities from multiple sources with hierarchical agglomerative clustering. In *Proceedings of the IC3K*. SCITEPRESS, 40–50. DOI: https://doi.org/10.5220/0010649600003064

[44] Alieh Saeedi, Eric Peukert, and Erhard Rahm. 2018. Using link features for entity clustering in knowledge graphs. In *The Semantic Web - Proceedings of the 15th International Conference, ESWC 2018, Proceedings (Lecture Notes in Computer Science, Vol. 10843)*. Springer, 576–592. DOI: https://doi.org/10.1007/978-3-319-93417-4_37

[45] Alieh Saeedi, Eric Peukert, and Erhard Rahm. 2020. Incremental multi-source entity resolution for knowledge graph completion. In *Proceedings of the European Semantic Web Conference*. Vol. 12123, Springer, 393–408. DOI: https://doi.org/10.1007/978-3-030-49461-2_23

[46] John R. Talburt and Yinle Zhou. 2013. A practical guide to entity resolution with OYSTER. In *Handbook of Data Quality, Research and Practice*. Shazia W. Sadiq (Ed.), Springer, 235–270. DOI: https://doi.org/10.1007/978-3-642-36257-6_11

[47] R. Endre Tarjan. 1974. A note on finding the bridges of a graph. *Inform. Process. Lett.* 2, 6 (1974), 160–161. DOI: https://doi.org/10.1016/0020-0190(74)90003-9

[48] André Valdestilhas, Tommaso Soru, and Axel-Cyrille Ngonga Ngomo. 2017. CEDAL: Time-efficient detection of erroneous links in large-scale link repositories. In *Proceedings of the International Conference on Web Intelligence*. ACM, 106–113. DOI: https://doi.org/10.1145/3106426.3106497

[49] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *Proceedings of the 6th International Conference on Learning Representations, ICLR 2018*. OpenReview.net. Retrieved from https://openreview.net/forum?id=rJXMpikCZ

[50] Ruixuan Xiao, Yiwen Dong, Junbo Zhao, Runze Wu, Minmin Lin, Gang Chen, and Haobo Wang. 2023. FreeAL: Towards human-free active learning in the era of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023*. Houda Bouamor, Juan Pino, and Kalika Bali (Eds.), Association for Computational Linguistics, 14520–14535. DOI: https://doi.org/10.18653/V1/2023.EMNLP-MAIN.896

[51] Zeyu Zhang and Yulong Pei. 2021. A comparative study on robust Graph Neural Networks to structural noises. Retrieved from https://arxiv.org/abs/2112.06070

[52] Zhanke Zhou, Jiangchao Yao, Jiaxu Liu, Xiawei Guo, Quanming Yao, LI He, Liang Wang, Bo Zheng, and Bo Han. 2023. Combating bilateral edge noise for robust link prediction. In *Advances in Neural Information Processing Systems 36: Proceedings of the Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023*. Retrieved from http://papers.nips.cc/paper_files/paper/2023/hash/435986a8cc3e0667648df5d1c2d55c83-Abstract-Conference.html

Appendix

A Linkage Configuration

To compute the similarity graph, we utilize the following configurations shown in Table 6.

Table 6. Overview of Linkage Configurations of Dexter, MusicBrainz, and WDC

Dataset	Dexter	MusicBrainz	WDC
Attributes	Heterog. key-value pairs	Artist, title, album, year, length, language, number	title, description, brand, Capacity, Manufacturer, Spindle Speed
Blocking Key	manufacturer name, model number	preLen1(album)	-
Similarity Function	Trigram(model names,product code, sensor type), Euclid(opt./digital zoom,camera dim.,price,weight, resolution	Trigram(title)	Levenshtein, Jaccard, Jaccard with inner Levenshtein, token overlap, token containment on string attributes, normalized absolute difference on numerical attributes

Received 22 August 2024; revised 16 March 2025; accepted 2 May 2025