Efficient Entity Resolution on Heterogeneous Records

Yiming Lin, Hongzhi Wang, Jianzhong Li, and Hong Gao

Abstract—Entity resolution (ER) is the problem of identifying and merging records that refer to the same real-world entity. In many scenarios, raw records are stored under heterogeneous environment. Specifically, the schemas of records may differ from each other. To leverage such records better, most existing work assume that schema matching and data exchange have been done to convert records under different schemas to those under a predefined schema. However, we observe that schema matching would lose information in some cases, which could be useful or even crucial to ER. To leverage sufficient information from heterogeneous sources, in this paper, we address several challenges of ER on heterogeneous records and show that none of existing similarity metrics or their transformations could be applied to find similar records under heterogeneous settings. Motivated by this, we design the similarity function and propose a novel framework to iteratively find records which refer to the same entity. Regarding efficiency, we build an index to generate candidates and accelerate similarity computation. Evaluations on real-world datasets show the effectiveness and efficiency of our methods.

Index Terms—Data source, data integration, entity resolution

1 Introduction

1.1 Motivation

Entity Resolution (ER) is the process of identifying and merging records that refer to the same real-world entity across different data sources. It is a crucial step for data cleaning and data integration.

As surveyed in [1], due to the success of social and Semantic Web applications, as well as the establishment of standards and best practices for publishing and exchanging data on the Web, a large and quickly growing volume of heterogeneous datasets has become available on the Web.

We address two motivations for ER on heterogeneous data. To find value from heterogeneous data, one way is to analyze heterogeneous records directly; the other is to integrate heterogeneous data under a united interface and make data analysis over it.

On one hand, many scenarios require us to analyze data directly under heterogeneous settings. For example, although users might be the same individuals in various platforms, their profiles and behaviors could differ greatly. Product recommendations can be improved if more accurate and complete user behaviors are captured, which benefits from entity resolution to distinguish user behaviors under different platforms. As another example, the house information could be shared by two or more housing companies. Entity resolution across various company websites could help make a better decision. However, most existing techniques studied ER on records under a predefined schema [2].

On the other hand, another fashion to leverage such heterogeneous data falls in the scope of data integration: transform those data under a media schema and provide a united surface for users. However, in many scenarios, the lineage of such records comes from heterogeneous sources, the schemas of which could vary from source to source. To convert records under different schemas to those under a predefined schema, two steps are often required, i.e., schema matching and data exchange. In the conventional framework shown in Fig. 1c, schema matching and data exchange are performed to resolve semantic heterogeneity and transform records to those under a predefined schema, alleviating the need for extensive approaches of ER. Even though such framework works for many scenarios, it would *lose information* in some cases, which is useful or even crucial to entity resolution. Information loss arises from the difference of information content in target schema and source schemas, which brings about both false positive and false negative errors. We use an example to illustrate it.

Example 1. Consider a company with three types of customer records shown in Fig. 1, each of which owns its schema, respectively. Given a target schema {{name},{position}, {addr},{city}} and schema matchings, we execute data exchange to convert the instances from source schema to the target schema shown in Fig. 1b. Note that ground truth is not required for our approach, but only for illustration.

Now consider records under the target schema. The similarity between r_7 and r_8 is high under whatever similarity metrics, such as edit distance, Jaccard similarity, etc. However, they do not refer to the same entity. It is infeasible to correct this false positive in target schema. Instead, if we explore information from records under source

• *The authors are with the Harbin Institute of Technology, Harbin 150006, China.*
E-mail: YimingLin_0426@hotmail.com, {wangzh, lijzh, honggao}@hit.edu.cn.

Manuscript received 25 July 2018; revised 26 Nov. 2018; accepted 30 Jan. 2019. Date of publication 7 Feb. 2019; date of current version 1 Apr. 2020. (Corresponding author: Hongzhi Wang.)
Recommended for acceptance by G. Li.
Digital Object Identifier no. 10.1109/TKDE.2019.2898191

1041-4347 © 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: Renmin University. Downloaded on July 18,2022 at 09:11:56 UTC from IEEE Xplore. Restrictions apply.

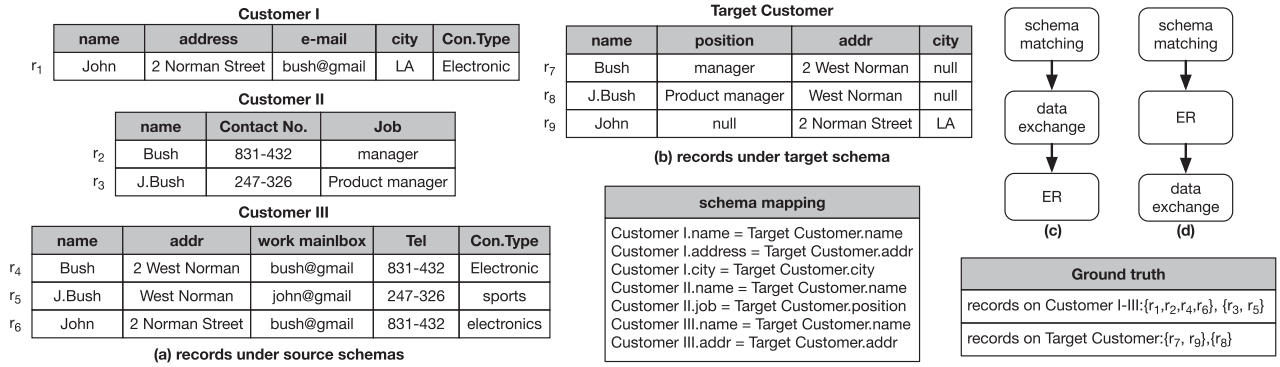


Fig. 1. Motivating example: A customer mapping scenario.

schemas, we find that r_7 is the join result of r_2 and r_4, and r_8 is the join result of r_3 and r_5. The information useful for entity resolution, e − mail, Tel/Contact No and Con.Type(Consumption type) are dropped during schema matching. Based on these distinguished information, we could infer that r_7 and r_8 refer to different entities.

Next consider false negative error. r_7 and r_9 share few similar fields under target schema but they refer to the same entity. As we observe records under source schema, r_9 is the join result of r_1 and r_6. For r_7 and r_9, we examine e-mail and TeL/Contact No, which is identical, and their Con.Type is very similar. Thus, it is possible to draw the conclusion that they refer to the same entity.

When the difference of information between source schema and target schema is distinguished, information loss is an inevitable obstacle for entity resolution on records under target schema. We use two scenarios to clarity that it is common in practice.

- When executing schema matching, we often match multiple source schemas to one target schema. Thus, the number of attributes in source schema is greater than that in target schema.
- Target schema is often designed for specific goals, which does not necessarily involve highly distinguished attributes.

To our best knowledge, none of existing technique of ER on homogeneous records or its transformation could solve this problem. When the lost information is discriminative, it would probably lower the quality of results for ER on records under target schema. We highlight that the impaired quality resulted from information loss *CANNOT* be offset by ER over homogeneous records.

Motivated by Example 1 and the above discussions, we seek to leverage sufficient information from heterogeneous sources to improve ER instead of only the attributes in target schema. Thus, we propose a new framework as shown in Fig. 1d. Given the schema matchings between source schemas and target schema, we perform ER on heterogeneous records directly. Finally, data exchange is implemented to convert heterogeneous records with entity labels to a predefined schema. Both frameworks shown in Figs. 1c and 1d finally generate labeled records under same schema.

1.2 Challenges for Heterogeneous Entity Resolution

Even though heterogeneous ER brings some advantages, new challenges are posed.

First, two records referring to the same real-world entity may share few attributes, leading to low similarity, since they describe different aspects of one entity. As an example, consider r_1 and r_2 in Fig. 1. Ground truth indicates that they refer to the same entity, but they only share one attribute name and the other six attributes are different. Thus, their similarity is low. We capture this phenomenon as *description difference*, which makes it non-trivial to match such heterogeneous record pairs correctly.

Second, when comparing two records, lack of schema matching *between source schemas* makes similarity computation difficult. For example, for r_1 and r_4, since we do not know whether Customer I.email and Customer III.work mailbox refer to the same attribute, it is hard to compare corresponding attribute values directly. We capture this challenge as *heterogeneous schema*. None of similarity function designed for records under identical schema [2] can be used to compute the similarity of two heterogeneous records directly.

To our best knowledge, the critical feature *description difference* is first defined by us. Furthermore, none of existing work on ER could handle above two challenges at the same time. Most work assumes that schema matching and data exchange have been accomplished to make the records from different sources be compared in a uniform manner [3], [4], [5].

There are several work about ER over heterogeneous records, machine learning based techniques [6], [7], human-based approaches [8], [9], blocking methods over heterogeneous information spaces, [10], [11], [12], etc. In Related work part (Section 2), we give a detailed discussion and show that they can not handle the above two challenges very well.

1.3 HERA

To solve the above two challenges, we propose **H***eterogeneous* **E***ntity* **R***esolution* **A***lgorithm* (HERA). HERA could handle records with various data types, such as string data, numeric data, etc. and view the similarity metric of corresponding data type as a black-box, which permits extensive ER resolution. Regarding *description difference*, HERA uses a compare-and-merge mechanism to iteratively find and merge similar records. For *heterogeneous schema*, HERA could compute the similarity of two records without any priori schema matchings. Furthermore, HERA can generate some high-reliable schema matchings to help record similarity computation. We also take the efficiency issue into account and design an efficient index. Based on it, we derive a tight upper bound and lower bound of record similarity to generate candidates within linear time. Leveraging index,

Authorized licensed use limited to: Renmin University. Downloaded on July 18,2022 at 09:11:56 UTC from IEEE Xplore. Restrictions apply.

the time complexity of record similarity computation is reduced by three orders of magnitude comparing with a basic nest-loop method and two records merging can be accomplished within logarithmic steps. Finally, we show the optimality of HERA in terms of the number of record comparisons.

1.4 Contributions

In summary, we make the following contributions:

- We propose a novel and extensive framework for ER under heterogeneous environment and point out two new challenges, among which *description difference* is first addressed by us.
- To tackle description difference and support such framework, we define a *super record* as the merged record of those referring to one entity. It provides more evidence for entity resolution on records with heterogenous schemas.
- Based on the framework, we propose HERA, a comprehensive algorithm involving a set of techniques, to handle ER on heterogenous records. Specifically, we propose an iterative compare-and-merge mechanism, bipartite-based heterogenous record matching approach and schema matching prediction method based on probabilistic majority voting. These novel techniques ensure the effectiveness of HERA, and we prove the optimality of HERA in terms of the number of record comparisons.
- Taking efficiency into consideration, we designed an index to support HERA to ensure a fast execution. Furthermore, we developed several optimization techniques with theoretical guarantee to further speed up HERA.
- We conduct a comprehensive evaluation of our techniques against real-world data sets. Our experimental results show that HERA achieves high performance and could collect valid positive evidence to improve the results of ER with a predefined schema significantly.

In the rest of the paper, we formally formulate ER on heterogeneous records and give an overview in Section 3. Section 2 discusses the related work. In Section 4 we design the index and propose approaches for record similarity computation in Section 5. Section 6 presents the overall solution for ER. Finally, in Section 7 we reports the experimental results with analysis; and Section 8 concludes the whole paper.

2 Related Work

Entity resolution is long-studied area, and a variety of methods for solving the ER problem have been proposed in literature. As surveyed in [2], [13], [14], [15], [16], most prior works focused on ER on records under a predefined schema. However, in many scenarios, information loss due to schema matching, especially the loss of some high identifying attributes, can lower the quality of ER significantly. It is the case that can hardly be handled by those ER techniques executing on homogeneous records.

There are several learning based techniques to solve ER under heterogeneous settings [6], [7]. [6] actually explore the semantic heterogeneity between field pairs but the records are still under homogeneous schema. [7] employed machine learning models for instance matching based on some similarity metrics of instances. [17] presented an unsupervised system that performs instance matching between entities in schema-free RDF files. They generated the training set without domain expertise or manually labeled samples. The full system is implemented as a sequence of components that can be iteratively executed to boost performance. All the above learning based techniques failed to capture *description difference*, since they rely on finding the similar instance values between instance pairs. However, *description difference* claims that two records referring to the same real-world entity could share few similar values, and two records referring to different entity could share many similar values.

Some human-based approaches [8], [9], [18], [19], [20], [21], [22] could be used to solve heterogeneous ER. However human-based method of ER in heterogeneous settings falls into low effectiveness since records satisfying *description difference* are difficult recognized by human, since such records belonging to one entity but share few attributes. (See r_1 and r_2 in Fig. 1) In addition, our method is orthogonal with their, i.e., crowd-sourcing based method could be embedded in our framework to improve the accuracy of similar record pair detection, and our method could provide some joint information to help disambiguate record pair for crowd-sourcing method.

Several blocking methods have been proposed for ER over heterogeneous information spaces. [10] defined a framework for blocking-based clean- clean ER that consists of two orthogonal layers: the effectiveness layer builds overlapping blocks with small likelihood of missed matches; the efficiency layer restricts the required number of pairwise comparison. They also proposed attribute clustering blocking and comparison scheduling techniques. [1] introduced the attribute-agnostic blocking methodology and made no use of schema information in the blocking step of ER, which could be applicable in heterogeneous settings with loose schema binding. Their work did not comprise the exact solution of record similarity computation. [12] empirically studied the behavior of existing blocking for algorithms for datasets exhibiting different semantic and structural characteristics, without proposing their own solution for heterogeneous ER.

Many state-of-the-art systems [23], [24], [25] resolved entities through an iterative process. Initially, they detect strongly similar entities, and then, use these resources as seeds for bootstrapping an iterative algorithm that detects new matches. This process is repeated until converging to a stable solution (i.e., no more matches are identified) or until no comparison exceeds the minimum similarity threshold. However, they can hardly tackle ER under heterogeneous settings, and also fail to capture description difference.

In summary, to overcome *information loss*, we proposed HERA, which can resolve both *description difference* and *heterogeneous schema*. Furthermore, as efficiency is always an important issue, we took it into consideration. All operations of HERA are supported by an efficient index, and we empirically show that HERA owns a high performance in real applications.

Authorized licensed use limited to: Renmin University. Downloaded on July 18,2022 at 09:11:56 UTC from IEEE Xplore. Restrictions apply.

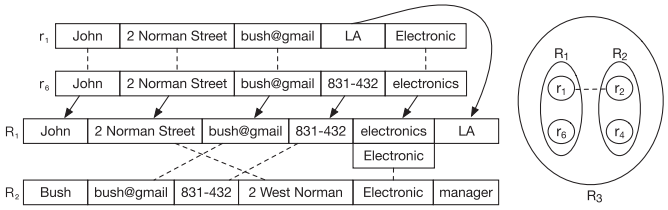


Fig. 2. Super record.

3 Problem Statement and Overview

In this section, we define the problem and related concepts in Section 3.1. In Section 3.2 we overview our solution.

3.1 Definitions

For a record set $\mathcal{R} = \{r_1, r_2, \ldots, r_n\}$ with heterogenous schemas, the schema of r_i is s_i with k_i attributes, $a_1^i, a_2^i, \ldots, a_{k_i}^i$. We denote by $a_t^i \approx a_l^j$ if a_t^i is mapped to a_l^j. The problem of entity resolution is defined as follows.

Definition 1. *Given a record set $\mathcal{R}$ with heterogenous schemas, identify and merge records that refer to the same real-world entity. (ER)*

As discussed in Section 1, ER over heterogenous data brings the new challenge of description difference. We say a pair of records satisfy *description difference*, if they refer to the same entity but share few attributes, and thus have a low similarity. Traditional approaches based on similarity functions can hardly handle such case. As an example, how to find r_1 and r_2, which belongs to same entity.

Facing this challenge, we propose *Compare-and-Merge* mechanism. That is, we merge the records that are determined as referring to the same entity into a *super record*, and perform entity resolution on super records instead of computing similarities between records directly. Super records extend the information of the entity it refers to. They provide more evidence to identify that a record refers to an entity. Continue with motivating example in Fig. 1. Ground truth indicates that r_1 and r_2 refer to the same entity. However, we cannot merge them directly since their similarity is low. As shown in Fig. 2, we observe that r_1 and r_6 are similar, thus we merge them into a super record R_1; r_2 and r_4 are similar, then we merge them into a super record R_2. Also, the similarity between R_1 and R_2 is high, thus we merge R_1 and R_2 into R_3. Now, r_1 and r_2 have been merged into one super record. As a result, super records could be used to resolve description difference. We highlight that we can not compute the similarity of each *single record* directly due to description difference. The structure of a super record is defined as follows.

Definition 2 (Super Record). *A super record $R = \{f_1^R, f_2^R, \ldots, f_{|R|}^R\}$, where f_i^R is the set of values corresponding to the ith field of R. $f_i^R = \{v_{1,i}^R, v_{2,i}^R, \ldots, v_{|f_i^R|,i}^R\}$, where $v_{j,i}^R$ is the jth value of f_i^R.*

In the remainder of this paper, if the location of a value or field in the record is not sensitive, then we abuse the notion. That is, let v^R and f^R be a value v and a field f of R.

Note that super record is a general representation of records. A single record is the simplest super record, where each field stores one value. Without loss of generality, if R_1 and R_2 are merged into R_3, we denote that by, $R_3 = R_1 \oplus R_2$. Next we illustrate such merging operation as follows.

Example 2. As shown in Fig. 2, to merge r_1 and r_6, first we merge corresponding fields of them, and then store multiple values for the same field (if any). For attribute Con.Type, r_1 and r_6 have different values {electronics}, {Electronic}, and we store both of them. For the other fields which are not matched, such as LA, 831 − 432, we add them into super record directly.

Next, we devise the similarity of two super records, which is non-trivial due to its sophisticated structure. A super record consists of several fields, thus we first deduce the similarity of field pairs, and then define the record similarity.

According to Definition 2, a field stores multiple values which refer to the same attribute of an entity. If two fields are similar, they must share some similar values. Thus we use the similarity of the most similar value pair to evaluate field similarity, which is defined as follows.

Definition 3 (Field Similarity). *Given two super records R_i, R_j, let $simf(f_k^{R_i}, f_l^{R_j})$ be the similarity of two fields $f_k^{R_i}, f_l^{R_j}$, $simv(v_{p,k}^{R_i}, v_{q,l}^{R_j})$ be the similarity of two values $v_{p,k}^{R_i}, v_{q,l}^{R_j}$. That is*

$$simf(f_k^{R_i}, f_l^{R_j}) = \max_{1 \le p \le |f_k^{R_i}|, 1 \le q \le |f_l^{R_j}|} \{simv(v_{p,k}^{R_i}, v_{q,l}^{R_j})\}.$$

Our approach could handle various data types, such as string data, numeric data, etc. And we view the corresponding similarity functions as black-box, which permits extensive ER solutions. Both typographical and semantics variations of data could be handled by our approach by involving special similarity function. For ease of illustration, in the motivating examples throughout our paper, we consider string data, which is most widely used, and take *Jaccard* as similarity metric.

Note that other string similarity functions, such as *Soft TF-IDF*, edit distance, etc, could be served as alternatives.

Since we perform ER on records with heterogeneous schema directly, when deciding the similarity of two records, the matchings of their corresponding schemas are unknown. In Section 5 we present the solution for finding such schema matchings. Here we assume that the matching information is available and give the following definition.

Definition 4 (Field Matching). *Given two super records R_i, R_j and a value similarity threshold ξ, if f^{R_i} and f^{R_j} are decided to be identical field of an entity, then $f^{R_i} \simeq f^{R_j}$, denoting that f^{R_i} and f^{R_j} are matched. Let $\mathcal{F}(i,j)$ be the field matching set. $\mathcal{F}(i,j) = \{(f^{R_i}, f^{R_j}) \mid f^{R_i} \simeq f^{R_j}, simf(f^{R_i}, f^{R_j}) \ge \xi\}$.*

Intuitively, the similarity between two records R_i and R_j is sum of the similarity between corresponding fields. In the case $|R_i| < |R_j|$, when most of attributes in R_i are similar to some attributes in R_j, R_i could be considered similar as R_j, even though it can hardly find similar attributes in R_i for some of attributes in R_j. With this consideration, to normalize the similarity between two records within [0,1] interval, we divide the sum by $min(|R_i|, |R_j|)$. Thus, the similarity between two records is defined as follows.

Definition 5 (Record Similarity). *Given two super records R_i, R_j, their similarity is defined as follows:*

Authorized licensed use limited to: Renmin University. Downloaded on July 18,2022 at 09:11:56 UTC from IEEE Xplore. Restrictions apply.

$$Sim(R_i, R_j) = \frac{\sum_{(f^{R_i}, f^{R_j}) \in \mathcal{F}(i,j)} simf(f^{R_i}, f^{R_j})}{\min(|R_i|, |R_j|)}. \quad (1)$$

We use an example to illustrate the similarity metrics above.

Example 3. Continuing with the motivating example, we consider two super records $R_1 = r_1 \oplus r_6$, $R_2 = r_2 \oplus r_4$, as shown in Fig. 2. First, regarding field similarity, for attribute Con.Type, $f_5^{R_1} = \{\{\text{Electronic}\}\}, \{\text{electronics}\}\}$, $f_5^{R_2} = \{\{\text{Electronic}\}\}$. Since simv($\{\text{Electronic}\}, \{\text{Electronic}\}$) > simv($\{\text{electronics}\}, \{\text{Electronic}\}$) (we set 2 q-grams), $simf(f_5^{R_1}, f_5^{R_2})$ = simv($\{\text{Electronic}\}, \{\text{Electronic}\}$) = 1.

The dotted lines represent field matchings. If we set $\xi = 0.35$, then the field matching set $\mathcal{F}(1,2) = \{(f_2^{R_1}, f_4^{R_2}), (f_3^{R_1}, f_2^{R_2}), (f_4^{R_1}, f_3^{R_2}), (f_5^{R_1}, f_5^{R_2})\}$. By accumulating field similarity of field pair in $\mathcal{F}(1,2)$ and dividing it by 6, we obtain the record similarity of R_1 and R_2, $Sim(R_1, R_2) = \frac{0.37+1.0+1.0+1.0}{6} = 0.56$.

3.2 Overview

In this part, we overview our solution for ER on heterogeneous records.

Due to *description difference* arisen from heterogeneous settings, those records could hardly be detected by batch processing, and we propose *merge-and-compare* mechanism to handle description difference. That is, we merge any similar records into a super record. When deciding whether two records r_i and r_j are similar, instead of computing their record similarity straightforwardly, we consider the similarity of their corresponding super records. If their super records are similar, we deem that r_i and r_j refer to the same entity. Then we merge their corresponding super records to find potential new matchings in later rounds.

To facilitate the above process, we adopt an iterative method to find and merge similar records. The stop condition is that the similarity of any two super records is below a predefined threshold. In each iteration, we first generate candidate record pairs and then verify them. These two steps are introduced as follows.

Candidate Generation. This step prunes dissimilar record pairs and ensures high similar record pairs as the candidates.

According to Definition 5, if two records are similar, they must share some similar values. With this consideration, we aim to obtain the record pairs with at least one shared similar value. To achieve this goal, we index similar value pairs, such that a set of record pairs sharing at least one similar value could be obtained as the candidate set within linear time. Then, to refine the candidate set, for each candidate record pair, we derive an upper bound and a lower bound of the record similarity with logarithmic steps. That is, given a record similarity threshold δ, for a record pair (R_i, R_j), if the upper bound of $Sim(R_i, R_j)$ is below δ, then we could prune this pair safely; if the upper bound of $Sim(R_i, R_j)$ is equal to its lower bound, then we could directly determine the record similarity without verification.

Verification. For each candidate record pair, we aim to verify it by judging whether its similarity exceeds the threshold.

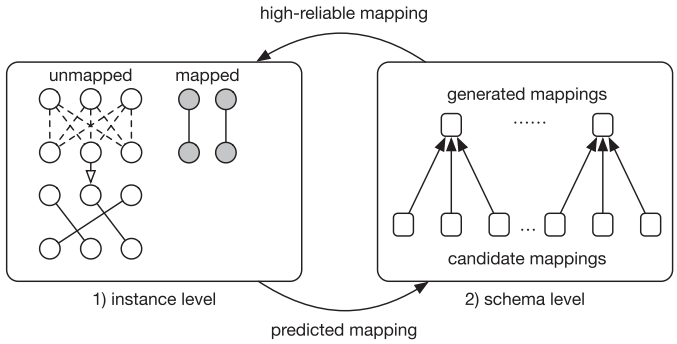


Fig. 3. Interaction of instance-based and schema-based method.

The core of this step is to compute the similarity of two heterogenous super records.

Such similarity computation is non-trivial due to twofold aspects: schema matchings between two records are missing, and the efficiency need to be taken into account owing to the complex structure of super records.

In absence of the schema matchings between records, we seek to derive similarity by detecting the similar degree of their values. Thus, we ignore the attributes information and treat a record as a set of values to derive record similarity, which is captured by *instance-based* method.

For two similar records, if two fields are high similar, their corresponding attributes could potentially match. Thus, a similar record pair would provide some predictions for corresponding schema matchings. We could use such predictions to generate some high-reliable schema matchings, which is captured by *schema-based* method.

With this consideration, we combine both *instance-based* and *schema-based* method for similarity computation because they can benefit each other. As shown in Fig. 3, instance-based method can provide predictions of schema matching, which leads schema-based method to determine more accurate matchings. With the high-reliable matchings generated by schema-based method, we could compute the similarity of two mapped fields directly, which can accelerate record similarity computation and improve the accuracy.

Next, we introduce these two methods briefly.

- *instance-based method:* The core part of record similarity computation is to determine the field matching set.

  According to Definition 4, basically, we first need to find all similar field pairs. This operation can be transformed to performing similarity join on two value sets of the record pair, which could be facilitated efficiently by index. Then, we model the problem of deciding field matching set as finding a maximum weight matching in bipartite graph, which is solved by Kuhn CMunkres [26] and sped up by graph simplification algorithms.
- *schema-based method:* Among predictions provided by instance-based method, one attribute may match multiple attributes. Consider a common assumption [27] that there are no redundant attributes in one schema. The determination of the matching relationship for such one-to-many matching is converted to the problem of finding the proper matching attribute from multiple choices. A probabilistic method based

Authorized licensed use limited to: Renmin University. Downloaded on July 18,2022 at 09:11:56 UTC from IEEE Xplore. Restrictions apply.

on majority vote could estimate the matching attribute with the upper bound of error probability. The details will be introduced in Section 5.2.

In summary, in the remainder of our paper, Section 4 presents the details of index building and candidate generation. Verification step would be described in Section 5.

4 Index

In Section 3.2, we overview the solution and show that index is required to achieve two goals. One is to generate candidate record pairs. The other is to accelerate record similarity computation. To achieve these goals, in this section, we introduce the index, whose optimization relies on the fact that the record similarity is essentially a combination of some value similarities. Thus, our goal is first to find similar value pairs between records efficiently leveraging index (Section 4.1). And then we describe how to reach the proposed optimization objectives. (Section 4.2).

4.1 Index Structure

Clearly, if two records share a lot of similar value pairs, they could be potentially similar. To generate candidate record pairs, we require to give a fast estimation of their similarity score using index efficiently. Thus, we index value pairs with similarity above the given value similarity threshold.

To begin with, for each value $v_{k,j}^{R_i}$ in $\mathcal{R}$, we assign it a unique label (rid, fid, vid), where $rid = i$, $fid = j$, $vid = k$. We assume that all the labels in our index begin from 1. In Fig. 2, the label of Electronic in R_1 is (1,5,2) and that of bush@gmail in R_2 is (2,2,1).

Definition 6 (Index). *Let $\mathcal{V}$ be the set of value pairs stored in index. For each value pair $(v_{k,p}^{R_i}, v_{l,q}^{R_j}) \in \mathcal{V}$ where $i \neq j$, we index it by assigning each value pair a label: $((i,p,k),(j,q,l))$. Next we execute the following operations:*

1) *For a value pair with label $((rid_1, fid_1, vid_1), (rid_2, fid_2, vid_2))$, exchange their locations to ensure $rid_1 < rid_2$;*
2) *Value pairs in $\mathcal{V}$ are sorted in the priority of rid_1, rid_2 and the value similarity. That is, value pairs are sorted according to rid_1 and rid_2 in ascending order. Those pairs with same rid_1 and rid_2 are sorted by the value similarity in descending order.*

Each value pair in index owns a pid, denoting pidth pair. In summary, for each value pair in index, we stored its pid, corresponding labels and value similarity.

As an example, the index for value pairs in motivating example (r_1 to r_6) is shown in Fig. 4. The part contained in the solid rectangle represents the index and contains totally 17 value pairs. Taking 13th value pair for instance, we store its pid as 13, label as ((4,1,1),(5,2,1)) and its similarity score as 0.83. From its label, we can find its corresponding record pair as (r_4, r_5). Noting that rid_1 and rid_2 (covered in dotted rectangle), rid_1 is sorted in ascending order. If two pairs share the same rid_1, they are sorted in ascending order by rid_2. In Fig. 4, 13th and 14th pair share the same rid_1 and rid_2. Thus, they are sorted in the descending order by similarity score.

To build the index, we require to find all similar value pairs, which satisfy both following two conditions:

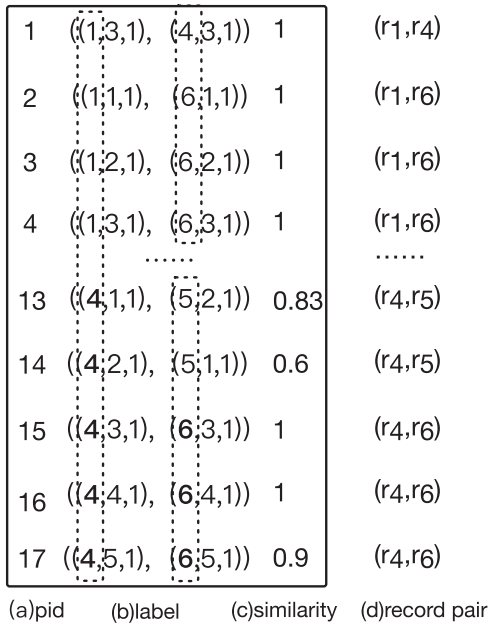


Fig. 4. Structure of index.

1) their value similarity is above ξ;
2) they belong to different records in $\mathcal{R}$;

This problem could be solved by *similarity join* [28] defined as follows.

Definition 7 (Similarity Join). *Given a record set $\mathcal{R}$, let $\mathbb{V}$ be value set present in $\mathcal{R}$, the result of* similarity join *on $\mathbb{V}$ is a set of value pairs $\mathcal{V}$*

$$\mathcal{V} = \{(v^{R_i}, v^{R_j}) \mid 1 \leq i, j \leq |\mathcal{R}|, i \neq j, simv(v^{R_i}, v^{R_j}) \geq \xi\}.$$

Note that similarity join is not limited to string data type. In technical report [29], we discuss the transformations of current *similarity join* techniques to apply to various data types.

Lei $\mathbb{V}_i$ and $\mathbb{V}_j$ be the set of values in records R_i and R_j, respectively. We denote by $\mathcal{V}_{ij}$ the results of similarity join on $\mathbb{V}_i$ and $\mathbb{V}_j$

$$\mathcal{V}_{ij} = \{(v^{R_i}, v^{R_j}) \mid simv(v^{R_i}, v^{R_j}) \geq \xi\}.$$

Note that $\forall i, j \in [1, |\mathcal{R}|]$, we have $\mathcal{V}_{ij} \subseteq \mathcal{V}$, i.e., we can obtain similar value pairs of any two records from index. Thus, similarity join on $\mathbb{V}$ requires only one pass. That is to say, the index could be built *off-line*, although the complexity of index construction is high.

Proposition 1. *Time complexity of index construction is $O(|\mathcal{V}|log|\mathcal{V}|) + O(|\mathcal{V}^2|)$.*

Proof is shown in technical report [29].

4.2 Index Operation

In this section, we describe the operations on index to reach the aforementioned optimization goals and then present the details of index update arising from record merging.

4.2.1 Candidate Generation

As discussed in Section 3.2, the goal of this operation is to filter dissimilar record pairs and determine the similarity of

Authorized licensed use limited to: Renmin University. Downloaded on July 18,2022 at 09:11:56 UTC from IEEE Xplore. Restrictions apply.

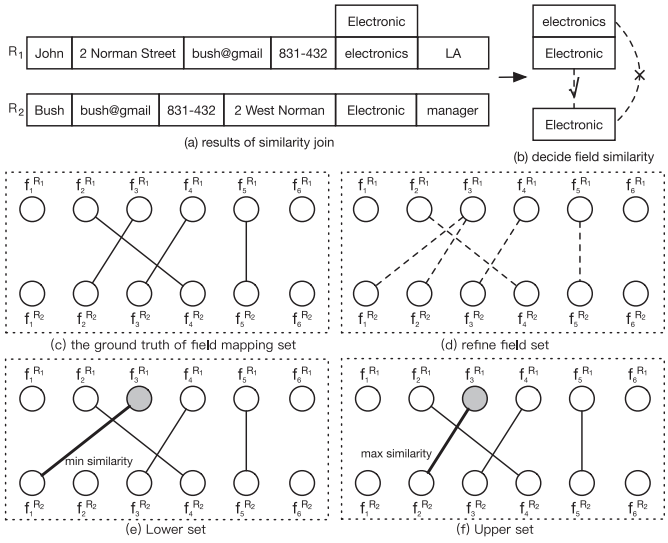


Fig. 5. Upper bound and lower bound.

some records directly. These two parts are excluded from the candidate set. Let $\mathbb{R}$ be the candidate set and $\mathbb{R}^{'}$ be the set of records whose similarity could be directly computed by index, respectively. Dissimilar pairs will not be considered furthermore, while the pairs in $\mathbb{R}^{'}$ are included in the final results without further computation.

The results of this step are *candidate* record pairs, which is for further verification. Accordingly, given a record similarity threshold δ, for (R_i, R_j), we deduce an upper bound and a lower bound of record similarity, denoted by $Up(R_i, R_j)$ and $Low(R_i, R_j)$.

If $Up(R_i, R_j) < \delta$, then (R_i, R_j) could be safely pruned since $Sim(R_i, R_j) < \delta$ must hold. If $Up(R_i, R_j) = Low(R_i, R_j)$, then we could directly determine $Sim(R_i, R_j) = Up(R_i, R_j)$.

According to Definition 5, the key of computing $Sim(R_i, R_j)$ is to decide a field matching set $\mathcal{F}_{ij}$, which consists of several matching field pairs following one-to-one matching. As an example in Fig. 5, we seek to deduce bounds of $Sim(R_1, R_2)$ and ξ is set as 0.3. We view each field as a point and Fig. 5c depicts the ground truth of field matching $\mathcal{F}_{12}$. Next we use $\mathcal{V}_{ij}$ to derive $Up(R_i, R_j)$ and $Low(R_i, R_j)$ as follows.

1) For each field pair, we store the value pair with the maximum similarity and delete others. We denote such field pair set after the above deletions by $\mathcal{V}_{ij}^{'}$, which is called the *refined field set*. As shown in Fig. 5b, for field pair $(f_5^{R_1}, f_5^{R_2})$, we retain $\{\{\mathsf{Electronic}\}, \{\mathsf{Electronic}\}\}$ and delete others since its similarity is maximum. Fig. 5d shows $\mathcal{V}_{12}^{'}$.
2) Regarding $\mathcal{V}_{ij}^{'}$, if a field is only contained in one field pair, we call it a *single field*; if a field is covered by more than one field pairs, we call it a *multiple field*. $f_3^{R_1}$ in Fig. 5d is a *multiple field* since it is covered by $(f_3^{R_1}, f_1^{R_2})$ and $(f_3^{R_1}, f_2^{R_2})$. Next consider two cases: For each *multiple field*, among the field pairs covering it, we retain the one with the maximum similarity and delete others. $\mathcal{V}_{ij}^{'}$ after such operations is called an *Upper set*, which is denoted by $\mathcal{US}_{ij}^{'}$ (shown in Fig. 5f); For each *multiple field*, among the field pairs covering it, we retain the one with minimum similarity and delete others. $\mathcal{V}_{ij}^{'}$ after such operations is called a *Lower set*, which is denoted by $\mathcal{LS}_{ij}^{'}$ (shown in Fig. 5e). Note that for each *multiple field*, among the field pairs covering it, only one pair is contained in $\mathcal{F}_{ij}$ since the pairs in $\mathcal{F}_{ij}$ follow one-to-one matching. And how to decide such a pair will be described in Section 5.1 in detail.

Thus, the set of field pairs that do not cover any *multiple field* is the intersection of $\mathcal{F}_{ij}$, $\mathcal{US}_{ij}^{'}$ and $\mathcal{LS}_{ij}^{'}$. Except them, for each *multiple field*, $\mathcal{US}_{ij}^{'}$ covers the pair with the maximum similarity and $\mathcal{LS}_{ij}^{'}$ covers the pair with the minimum similarity. Therefore, the sum of field similarities for the above three sets satisfies the following inequalities

$$\begin{aligned}\sum_{(f^{R_i}, f^{R_j}) \in \mathcal{LS}_{ij}^{'}} simf(f^{R_i}, f^{R_j}) &\leq \sum_{(f^{R_i}, f^{R_j}) \in \mathcal{F}(i,j)} simf(f^{R_i}, f^{R_j}) \\ &\leq \sum_{(f^{R_i}, f^{R_j}) \in \mathcal{US}_{ij}^{'}} simf(f^{R_i}, f^{R_j}).\end{aligned} \tag{2}$$

According to Definition 5, the upper bound and lower bound of $Sim(R_i, R_j)$ is

$$Up(R_i, R_j) = \frac{\sum_{(f^{R_i}, f^{R_j}) \in \mathcal{US}_{ij}^{'}} simf(f^{R_i}, f^{R_j})}{\min(R_i, R_j)} \tag{3}$$

$$Low(R_i, R_j) = \frac{\sum_{(f^{R_i}, f^{R_j}) \in \mathcal{LS}_{ij}^{'}} simf(f^{R_i}, f^{R_j})}{\min(R_i, R_j)}. \tag{4}$$

Typically, if *multiple field* does not exist in R_i or R_j, we have $\mathcal{F}_{ij} = \mathcal{US}_{ij}^{'}$ and $\mathcal{F}_{ij} = \mathcal{LS}_{ij}^{'}$. In this case, the lower bound is equal to upper bound, thus record similarity is exactly the upper or lower bound. For such record pairs, we exclude them from candidate set and merge them directly. The details of merge operation is presented in Section 4.2.2.

As an example in Fig. 5, $\mathcal{US}_{12}^{'} = \{(f_2^{R_1}, f_4^{R_2}), (f_3^{R_1}, f_2^{R_2}), (f_4^{R_1}, f_3^{R_2}), (f_5^{R_1}, f_5^{R_2})\}$, $\mathcal{LS}_{12}^{'} = \{(f_2^{R_1}, f_4^{R_2}), (f_3^{R_1}, f_1^{R_2}), (f_4^{R_1}, f_3^{R_2}), (f_5^{R_1}, f_5^{R_2}),\}$. $f_3^{R_1}$ is the only *multiple field*. $\mathcal{US}_{12}^{'}$ contains the field pair $(f_3^{R_1}, f_2^{R_2})$ with the maximum similarity, i.e., $(\mathsf{bush@gmail}, \mathsf{bush@gmail})$. While $\mathcal{LS}_{12}^{'}$ contains $(\mathsf{bush@gmail}, \mathsf{Bush})$ with minimum similarity. Thus, $Up(R_1, R_2) = \frac{0.37+1+1+1}{6} = 0.56$, $Low(R_1, R_2) = \frac{0.37+0.33+1+1}{6} = 0.45$.

Next we illustrate how to compute $Up(R_i, R_j)$ and $Low(R_i, R_j)$ through index in Algorithm 1. For convenience of expression, each value pair is expressed as $((rid_1, fid_1, vid_1), (rid_2, fid_2, vid_2))$ and the pidth value pair is denoted by $pair^{pid}$, whose similarity is $simv_{pid}$.

In Algorithm 1, we set two arrays sim and id. $sim[i][j]$ is to record the maximum similarity of a field pair (f^{R_i}, f^{R_j}) so far and id is to record the pid of corresponding value pair (Line 1). $flagU$, $flagL$ and pre is used to find $\mathcal{US}_{ij}^{'}$ and $\mathcal{LS}_{ij}^{'}$ (Line 2). First, we seek to find $\mathcal{V}_{ij}^{'}$ (Lines 4-8). $binary\_search\_l(1, |\mathcal{V}|, i)$ searches the first and the last occurrences of i as the value of rid_1 between the first and the $|\mathcal{V}|$th entries in $\mathcal{V}$. This function is implemented by binary search since $\mathcal{V}$ is sorted by rid_1. $binary\_search\_r(k, l, j)$ searches the first and the last occurrences of j as the value of rid_2 in between kth and lth entries in $\mathcal{V}$. This function is also implemented by binary

Authorized licensed use limited to: Renmin University. Downloaded on July 18,2022 at 09:11:56 UTC from IEEE Xplore. Restrictions apply.

search since $\mathcal{V}$ is sorted by rid_2 when rid_1 of value pair is same. After that, the set of value pairs whose $pid \in [p, q]$ corresponds to $\mathcal{V}_{ij}$. Next, for each field pair, we only store the value pair with maximum similarity and delete others to obtain $\mathcal{V}_{ij}^{'}$ (Lines 7-10).

Algorithm 1. Cal_bound(i,j)

Input: $\mathcal{V}, i, j$;
Output: $Up(R_i, R_j)$, $Low(R_i, R_j)$;
1: $sim \leftarrow 0$, $id \leftarrow 0$; // Sim to record similarity, id to record corresponding pid;
2: $flagU \leftarrow false$, $flagL \leftarrow false$, $pre \leftarrow 0$;
3: // To find $\mathcal{V}_{ij}^{'}$;
4: $(k, l) \leftarrow binary\_search\_l(1, |\mathcal{V}|, i)$;
5: $(p, q) \leftarrow binary\_search\_r(k, l, j)$;
6: **for** $p \leq pid \leq q$ **do**
7: **if** $sim_{pid} > sim[fid_1][fid_2]$ **then**
8: $\mathcal{V}_{ij}^{'} \leftarrow \mathcal{V}_{ij}^{'} \cup pair^{pid}$, $\mathcal{V}_{ij}^{'} \leftarrow \mathcal{V}_{ij}^{'} \setminus pair^{id[fid_1][fid_2]}$, $sim[fid_1][fid_2] \leftarrow sim_{pid}$, $id[fid_1][fid_2] \leftarrow pid$;
9: **end if**
10: **end for**
11: //So far we have computed $\mathcal{V}_{ij}^{'}$. Next find $\mathcal{US}_{ij}^{'}$ and $\mathcal{LS}_{ij}^{'}$;
12: **for** each $pair^{pid} \in \mathcal{V}_{ij}^{'}$ **do**
13: **if** !$flagU[rid_1][fid_1]$ **then**
14: $flagU[rid_1][fid_1] \leftarrow true$, $\mathcal{US}_{ij}^{'} \leftarrow pair^{pid}$;
15: **end if**
16: **if** !$flagL[rid_1][fid_1]$ **then**
17: $flagL[rid_1][fid_1] \leftarrow true$;
18: **if** $pre \neq 0$ **then**
19: $\mathcal{LS}_{ij}^{'} \leftarrow pair^{pre}$;
20: **end if**
21: **end if**
22: $pre \leftarrow pid$;
23: **end for**
24: Compute $Up(R_i, R_j)$, $Low(R_i, R_j)$ according to Equations (3) and (4).
25: **return** $Up(R_i, R_j)$, $Low(R_i, R_j)$;

Then, we find $\mathcal{US}_{ij}^{'}$ and $\mathcal{LS}_{ij}^{'}$ (Lines 12-23). For each field in R_i, we select the field pair covering it with maximum similarity. Since value pairs are sorted in descending order of its similarity value when rids are same, value pair we first meet is the one with maximum similarity (Lines 12-15). For each field in R_i, we select the pair covering it with minimum similarity (Lines 16-23). Finally, $Up(R_i, R_j)$ and $Low(R_i, R_j)$ are computed (Line 24).

Example 4. In Fig. 4, the index is enveloped in a solid rectangle. If we attempt to find $V_{46}^{'}$, we first locate 4 using binary search as rid_1 for each value pair, which are enveloped in a dotted rectangle. We find 4 (in bold) appears from 13th to 17th value pair. Then we binary search 13th to 17th value pairs to find the entry with $rid_2 = 6$. Finally, we find three value pairs, $\{((4,3,1),(6,3,1)),((4,4,1),(6,4,1)),((4,5,1),(6,5,1))\}$. Next we deduce $Up(r_4, r_6)$ and set the record similarity threshold δ as 0.5. $Up(r_4, r_6) = \frac{1+1+0.9}{\min(5,5)} = 0.58$, $Low(r_4, r_6) = \frac{1+1+0.9}{\min(5,5)} = 0.58$, both of which are same because there are no *multiple field* for (r_4, r_6). Thus, (r_4, r_6) is not a candidate pair since its similarity is computed directly by index.

In total, $\binom{2}{6} = 15$ record pairs are generated from 6 records. Using the upper and lower bound constraint, there is only one candidate, $\{(r_2, r_4)\}$.

Proposition 2. Candidate generation *requires* $O(|\mathcal{V}|)$ *steps*.

We show the proof in technical report [29].

4.2.2 Index Maintenance

To resolve *description difference* discussed in Section 1, once two records are judged to be similar, we merge them into a super record. Since merging two records would change the labels of corresponding value pairs, such operation may involve further modification of index structure and is not trivial. Thus, in this section, we discuss how to maintain index efficiently when merging two records.

Without loss of generality, we consider merging R_i and R_j into R_k. The process is as follows.

1) *merge:* We use the union-find structure [30] to maintain rid (The functions of unoin and find are shown in [29]), i.e., $\mathsf{k} = \mathsf{unoin}(\mathsf{i}, \mathsf{j})$. Next, for each field pair $(f^{R_i}, f^{R_j}) \in \mathcal{F}_{ij}$ (see Definition 4), we merge f^{R_i} and f^{R_j} into one field f^{R_k} and then merge their corresponding values. Thus, each value in R_k would be assigned a new label.
2) *delete:* For those value pairs between R_i and R_j, we delete them in index to satisfy the constraint in Definition 6, i.e., two values belongs to different records.
3) *update:* For each value contained in R_i or R_j, we update its label in the index and adjust the order of corresponding value pair to satisfy the second constraint in Definition 6.

Please note that merging two records would simplify the index, which could potentially accelerate the process of candidate generation and index maintenance in further entity resolution processing. Such simplification would not affect the correctness of *candidate generation* and *merge* operations in later iterations. We highlight this property as follows.

Proposition 3. $\forall i, j \in [1, |\mathcal{R}|]$, *let* $f(i) = \mathsf{find}(i)$, $f(j) = \mathsf{find}(j)$, *and* $\mathcal{V}_{f(i)f(j)} \subseteq \mathcal{V}$ *always holds.*

For any record R_i, $f(i)$ is the rid of its corresponding super record. Proposition 3 indicates that for two super records which are merged by arbitrary records, their similar value pairs could be obtained through index, thus ensuing the correctness of candidate generation and merge operation.

Example 5. We consider merging r_1 and r_6 shown in Fig. 2. We assume that $\mathsf{1} = \mathsf{union}(\mathsf{1}, \mathsf{6})$, i.e., $R_1 = r_1 \oplus r_6$.

First, we delete value pairs between r_1 and r_6. As shown in Fig. 6, four value pairs are removed. Next, for each value contained in r_1 or r_6 (all the $rids$ of such values are in bold), we update its label. Take the value electronics with label (6,5,1) for example. After merging into R_1, its label updates to (1,5,1).

Proposition 4. *The computational complexity of index maintenance for merging two super records is* $O(|\widehat{\mathcal{V}}_{ij}|log|\mathcal{V}|)$.

$|\widehat{\mathcal{V}}_{ij}|$ is number of value pair w.r.t R_i or R_j in index. (In Example 5, it is 9) The proof is shown in technical report [29].

Authorized licensed use limited to: Renmin University. Downloaded on July 18,2022 at 09:11:56 UTC from IEEE Xplore. Restrictions apply.

((1,3,1), (4,3,1)) → ((1,3,1), (4,3,1))
((1,1,1), (6,1,1)) → ~~((1,1,1), (6,1,1))~~
((1,2,1), (6,2,1)) → ~~((1,2,1), (6,2,1))~~
((1,3,1), (6,3,1)) → ~~((1,3,1), (6,3,1))~~
((1,5,1), (6,5,1)) → ~~((1,5,1), (6,5,1))~~
((2,2,1), (6,4,1)) → ((2,2,1), (1,4,1))
((4,3,1), (6,3,1)) → ((4,3,1), (1,3,1))
((4,4,1), (6,4,1)) → ((4,4,1), (1,4,1))
((4,5,1), (6,5,1)) → ((2,2,1), (1,5,1))

Fig. 6. Label update by merge operation.

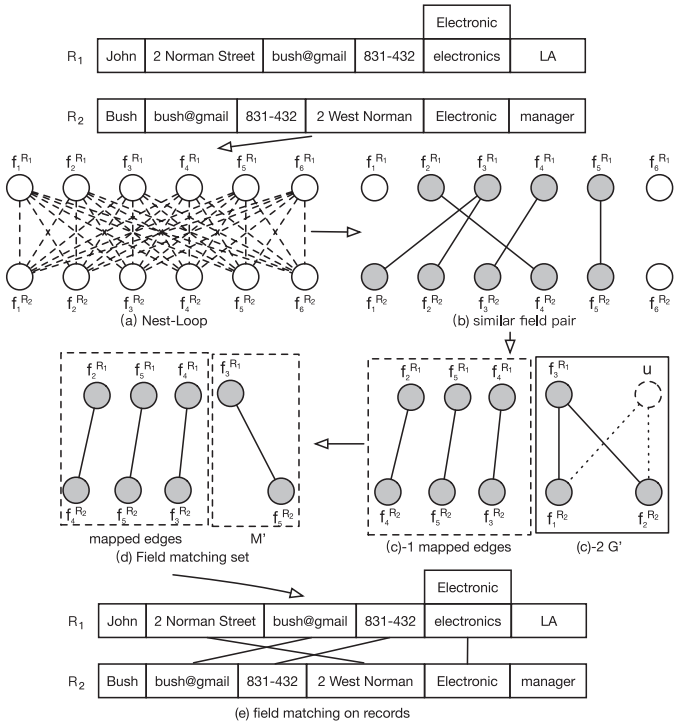


Fig. 7. Instance-based method.

5 Verification

In this section, we compute record similarity to verify candidate record pairs as discussed in Section 4. To achieve this goal, we proposed *instance-based* (Section 5.1) and *schema-based* methods (Section 5.2), respectively.

5.1 Instance-Based Method

In absence of schema matchings, we treat each record as a set of values ignoring the information of attributes. If two records are similar, they must share some similar values. We compute record similarity by detecting and quantifying the similar degree of value pairs between records, which is captured by *instance-based* method.

For instance-based method, following two problems are to be solved.

1) How to compute record similarity accurately without the information of schema matchings?
2) How to execute computation efficiently due to the sophisticated structure of super record?

Given two records R_i, R_j and a value similarity threshold ξ, according to Definition 5, first we need to find a field matching set $\mathcal{F}_{ij}$, and then accumulate the similarity values of each field pair covered by $\mathcal{F}_{ij}$ to obtain $Sim(R_i, R_j)$.

Recall that a field matching set (see Definition 4) consists of such field pairs that satisfy the following two conditions: (1) their field similarity is no less than ξ; (2) they refer to the same attribute of an entity. We call a field pair as a *similar field pair*, if its similarity is no less than ξ.

Thus, we proceed the process as two steps for record similarity computation in instance-based method: find field matching set and then compute similarity.

Step 1. Find Field Matching Set. According to Definition 4, basically, we first need to find all field pairs whose similarity is above ξ.

To reach the goal, a naive approach *nest-loop* (shown in Fig. 7a) needs to compute the similarity of all value pair $(v_{p,k}^{R_i}, v_{q,l}^{R_j})$ and then calculate field similarity for each field pair. Such a process requires four loops on variable p, k, q, l, which is time-consuming.

Recall that a field similarity (see Definition 3) is essentially a value similarity. Instead of nest-loop, we present an efficient algorithm inspired by *similarity join*. We view values in R_i and R_j as two value sets, respectively. Similarity join is performed on those two sets to find all value pairs whose similarity exceeds ξ. Based on similar value pair set, we can obtain corresponding *similar field pairs* very fast. In our approach, we do not actually execute similarity join for each record pair, which is quite expensive. Instead, we can obtain the similar field pairs of a record pair leveraging index directly as follows.

Recall that in Section 4.2.1, $\mathcal{V}_{ij}$ is the result of similarity join on $\mathbb{V}_i$ and $\mathbb{V}_j$, which are the set of values in R_i and R_j. $\mathcal{V}_{ij}^{'}$ is refined field set of $\mathcal{V}_{ij}$: for each field pair, keep the value pair with the maximum similarity and delete others, which corresponds to the definition of *field similarity*. (see Definition 3) Thus, $\mathcal{V}_{ij}^{'}$ is exactly the set of all *similar field pairs*. By Algorithm 1 (lines 1-8) we can obtain $\mathcal{V}_{ij}^{'}$ through index within $O(|\widehat{\mathcal{V}}_{ij}| + log|\mathcal{V}|)$ steps.

As an example shown in Fig. 7, $R_1 = r_1 \oplus r_6$, $R_2 = r_2 \oplus r_4$, we seek to compute $Sim(R_1, R_2)$. Each field is modeled as a point. The dotted line in Fig. 7a represents each comparison of field pair by nest-loop. The solid line in Fig. 7b expresses the *similar field pair*. From here we observe that *similarity join* reduced comparisons significantly.

After *similar field pairs* are prepared, we seek to select some pairs to generate a field matching set. Next we propose a graph-based method to achieve this goal.

A Graph-based method. The pairs in the field matching set must follow one-to-one matching, i.e., they could not share any field. The one-to-one matchings between two field sets may have multiple choices. We observe that a potential true field matching set would maximize $\sum_{(f^{R_i}, f^{R_j}) \in \mathcal{F}_{ij}} simf(f^{R_i}, f^{R_j})$. Thus, we formulate the problem of deciding field matching set into finding a maximum weighted matching in a bipartite graph as follows.

Authorized licensed use limited to: Renmin University. Downloaded on July 18,2022 at 09:11:56 UTC from IEEE Xplore. Restrictions apply.

Definition 8 (Field Matching Problem). *Given two records R_i, R_j and a value similarity threshold ξ, we construct an undirected graph $G(V, E)$, where $V = X \cup Y$ with $X \cap Y = \emptyset$ and $E \subseteq X \times Y$ as follows:*

Let a field be a node and a field pair be an edge.

1) $E = \{(f_k^{R_i}, f_l^{R_j}) \mid simf(f_k^{R_i}, f_l^{R_j}) \geq \xi\}$;
2) *Let V be the point set covered by E.* $X = \{f_k^{R_i} \mid f_k^{R_i} \in V, 1 \leq k \leq |R_i|\}$ *and* $Y = \{f_l^{R_j} \mid f_l^{R_j} \in V, 1 \leq l \leq |R_j|\}$;
3) *Each edge e_{kl} has a weight w_{kl}, which corresponds to $simf(f_k^{R_i}, f_l^{R_j})$.*

Then, we aim to find a matching $\mathcal{M}$ of maximum weight, i.e., to maximize $w(\mathcal{M}) = \sum_{e_{ij} \in \mathcal{M}} w_{ij}$.

Before discussing the solution, the following observations lead to chances of simplifying the graph, which can accelerate matching process significantly. For an edge e, if the degree of two vertices of e are both one, then e can be deleted from graph. We refer to such an edge e as a *mapped edge*, denoting that two vertices of e are decided to map and thus e can be removed to avoid further computation. Here we denote the degree of a point u by $d(u)$.

Graph Simplification. When constructing graph G, we count $d(u)$ for each $u \in V$. For each edge $e = (x, y)$, if $d(x) = 1$ and $d(y) = 1$, then we delete e and its endpoints x, y from G, i.e., $E = E \setminus \{e\}$, $X = X \setminus \{x\}$ (if $x \in X$), $Y = Y \setminus \{y\}$ (if $y \in Y$).

Let $G' = (X' \cup Y', E')$ be the simplified graph, and $\mathcal{E}$ be the set of deleted edges, i.e., *mapped edges*. As shown in Fig. 7, Fig. 7c1 represents the mapped edges. The solid points and lines in Fig. 7c2 denotes G'.

So far, the field matching problem has been converted to finding a *Maximum Weight Matching* on G', which can be solved by a well-known Kuhn CMunkres(KM) algorithm [26] (details of KM are shown in technical report [29]. Note that KM algorithm requires a complete bipartite graph, that is, $|X'| = |Y'|$. We can add dummy points and set the weight of their corresponding edges to be zero to satisfy the above condition. As shown in Fig. 7c, we add u to let $|X'| = |Y'|$ and the weight of edges connecting with x is 0. The output of KM on G' is a matching $\mathcal{M}'$ with maximum weight.

Finally, we have obtained $\mathcal{M}'$ and a set of mapped edge $\mathcal{E}$, which is shown in Fig. 7d. Next we show that $\mathcal{M}' \cup \mathcal{E}$ is the match of maximum weight in graph G.

Theorem 1. *Match of maximum weight in G, $\mathcal{M} = \mathcal{M}' \cup \mathcal{E}$.*

Theorem 1 indicates that the deleted edges are a part of optimal solution, and the simplification of graph will not affect the optimality of solution. We show the proof in technical report [29]. According to Definition 8, the field matching set of R_i and R_j corresponds to $\mathcal{M}$. Note that we are not approximating the maximal weighted matching. Instead, we are reducing the graph size for the maximal weighted matching by leveraging index and applying graph simplification, and the matching process is exact.

Step 2. Similarity Computation Finally, we accumulate similarity values of field pairs in $\mathcal{F}_{ij}$ and compute $Sim(R_i, R_j)$ according to Definition 5.

The time complexity of *instance-based* method is $O(log|\mathcal{V}|)+O(m^3)$ ($m = \max(|X'|, |Y'|)$), where $O(log|\mathcal{V}|)$ is computational steps of finding $\mathcal{F}ij$ through index and $O(m^3)$ is contributed by KM algorithm. Note that, although the complexity is cubic w.r.t. m, our experimental results in Section 7 show that in many real-world datasets m^3 is far less than $|\mathcal{V}|$ and *instance-based* method is very fast in practice.

The *instance-based* method has two results: record similarity and predicated schema matchings produced by KM algorithm (shown in Fig. 7d). Each time if two records are judged to be similar, then they introduce some new schema matching information. The next part *schema-based* method tends to leverage such information to decide the true schema matchings, which can accelerate similarity computation and improve accuracy.

5.2 Schema-Based Method

Schema-based method aims to collect the schema matching information generated by *instance-based* method to decide potential true schema matchings.

We observe that schema matching predictions provided by *instance-based* method for various record pairs may contradict each other. Consider motivating example in Fig. 1. We set $\xi = 0.3$, John in r_1 is similar to john@bush in r_5, indicating that Customer I.name$\approx$Customer III.e $-$ mail. However, John in r_1 is identical with John in r_6, indicating that Customer I.name$\approx$Customer III.name. For Customer I.name, we aim to determine the true matching among different providers. Under the assumption[27] that there are no redundant attributes in one schema, for an attribute, among its predicted matching attributes under one schema, only one is true.

Under schema s_i, for an attribute a_k^i, there are n predictions so far. We formulate n predications as n independent trials. Let x_i be the outcome of the ith trial and x^* be the true matching. Then we construct independent Bernoulli random variables $X_1, X_2, \ldots, X_n$ as follows: if $x_i \neq x^*$, let $X_i = 0$; if $x_i = x^*$, let $X_i = 1$.

Intuitively, among n trials, if a matching occurs more often, then it could more likely be the true one. Thus, we adopt a majority vote method to estimate the true matching. Specifically, among n trials, we take the outcome with the highest frequency as the true one. However, such a vote method may lead to errors, especially on the case when n is small. Accordingly, we derive the upper bound of error probability, denoted by UP_{error}.

Let $\widehat{x}$ be the estimated value by majority vote. We denote by $p = Pr(x = x^*)$, where p is a priori value obtained by training dataset. Then we give the upper bound as follows.

Theorem 2. $UP_{error} = e^{-\frac{n}{2p}(p-\frac{1}{2})^2}$

Proof 1. Theorem 2.

Let $X = \sum_{i=1}^{n} X_i$, then we have

$$\begin{aligned} Pr_{error} &= Pr(\widehat{x} \neq x^*) \\ &= 1 - Pr(\widehat{x} = x^*) \\ &\leq 1 - Pr(X > n/2) \\ &= Pr(X \leq n/2). \end{aligned}$$

Then, we use chernoff bound to devise an upper bound of $Pr(X \leq n/2)$. For ease of illustration, let $a = n/2$. For every $t > 0$, we have the following deductions:

Authorized licensed use limited to: Renmin University. Downloaded on July 18,2022 at 09:11:56 UTC from IEEE Xplore. Restrictions apply.

$$Pr(X \leq n/2) = Pr(e^{-tX} \geq e^{-ta}) \leq \frac{E[e^{-tX}]}{e^{-ta}} \quad (5)$$

$$= e^{ta}((1-p) + pe^{-t})^n \leq \left(\frac{e^{-\varrho}}{(1-\varrho)^{1-\varrho}}\right)^{np} \quad (6)$$

$$\leq e^{-\frac{\varrho^2}{2}np} = e^{-\frac{n}{2p}(p-\frac{1}{2})^2}. \quad (7)$$

Inequality (5) holds by applying Markov's inequality to e^{tX}. In inequality (6), $0 < \varrho < 1$, $t = -ln(1-\varrho)$. Equation (7) holds for the following deduction:

We only need to show $\frac{e^{-\varrho}}{(1-\varrho)^{1-\varrho}} < e^{-\frac{\varrho^2}{2}}$. We take logarithm on both sides and construct a function $f(\varrho) = -\varrho - (1-\varrho)ln(1-\varrho) + \varrho^2/2$. Since $f''(\varrho) < 0$ and $f'(0) = 0$, we have $f(\varrho) \leq 0$ w.r.t. $0 < \varrho < 1$.

Thus, the upper bound of Pr_{error}, $UP_{error} = e^{-\frac{n}{2p}(p-\frac{1}{2})^2}$. □

Given a error probability threshold ρ, if $UP_{error} < \rho$, we decide $\widehat{x}$ to be the true matching with the probability of $1 - UP_{error}$ and embed it into *instance-based* method in the following comparisons. That is, once a matching $(a_k^i \approx a_l^j)$ is determined to be true, in the later comparisons we can directly include corresponding field pair $(f_k^{R_i}, f_l^{R_j})$ into the field matching set.

As an example, suppose $p = 0.8, n = 10, \rho = 0.6$. We have $UP_{error} = 0.57$ and we decide $\widehat{x}$ as the true matching with the probability 0.43.

6 Overall Solution

In this section, we introduce our **H***eterogeneous* **E***ntity* **R***esolution* **A***lgorithm* in in Algorithm 2 and present some analysis.

First, we build index as Definition 6 (Line 1). Then we initialize a two-dimensional vis array to be false to record whether two records are compared. (Line 2) The stop$_c$ondition is that $\forall R_i, R_j \in \mathcal{R}, Sim(R_i, R_j) \geq \delta$. When no two records can be further merged, our algorithm stops (Line 3). Next, we perform Candidate generation to decide candidate record pairs, which is described in Section 4.2.1 (Line 4). During candidate generation, besides pruning dissimilar record pairs, we also compute the similarities of a record set directly, to generate $\mathbb{R}'$. For each record pair in $\mathbb{R}'$, we merge them directly (Lines 5-6). Next for each record pair in candidates set $\mathbb{R}$, if they are not compared so far, then we compute record similarity (using instance − based method in Section 5.1). (Lines 8-10) If R_i and R_j are similar, we execute schema − based method to use the predictions of schema matchings to decide true ones. (see Section 5.2). Then we merge two record using techniques in Section 4.2.2 (Lines 6-10). After a new super record is generated, we delete R_i and R_j and update the corresponding index. (Lines 11-19) Finally, for each record r_i, we use union − find to get the rid of its corresponding super record, which is used as its entity label (Lines 21-23). Note that we use *id* to assign each record a unique number by using some hash function, and $id[i]$ denotes the unique number of record R_i. We avoid taking the subscript as an id since two different records may share the same subscript in *HERA*.

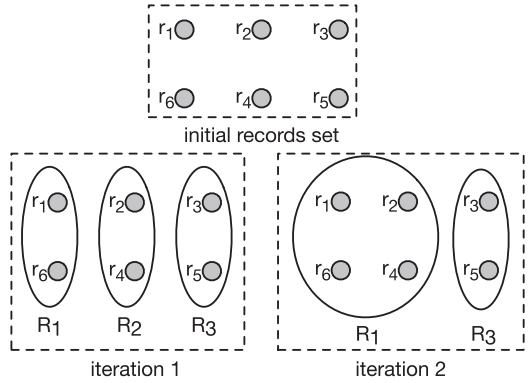


Fig. 8. Overall solution.

Algorithm 2. HERA

Input: Record set $\mathcal{R}$,
record similarity δ, string similarity ξ;
Output: $\mathcal{R}' : \{(r_i, eid) \mid r_i \in \mathcal{R}\}$ Record set with entity labels;

```
1: build index;
2: vis ← false;
3: while !stop_condition do
4:    Candidate generation;
5:    for each (R_i, R_j) ∈ ℝ' do
6:       merge(R_i, R_j);
7:    end for
8:    for each (R_i, R_j) ∈ ℝ do
9:       if vis[id_i, id_j] == false then
10:         Compute Sim(R_i, R_j);
11:         if Sim(R_i, R_j) ≥ δ then
12:            schema_based(R_i, R_j);
13:            merge(R_i, R_j);
14:            delete(R_i, R_j);
15:            update index;
16:            vis[id_i, id_j] = true;
17:         end if
18:      end if
19:   end for
20: end while
21: for r_i ∈ ℛ do
22:    ℛ' ← ℛ' ∪ (r_i, find(i));
23: end for
24: return ℛ';
```

Continue with motivating example. We set value similarity threshold $\xi = 0.5$ and record similarity threshold $\delta = 0.5$. $\mathcal{R} = \{r_1, r_2, \ldots, r_6\}$ and Fig. 8 shows the overall process of HERA. In Iteration 1, we merge three similar record pairs into three super records R_1, R_2 and R_3, respectively. It is worth to note that the index is updated arisen from merge operation. In iteration 2, we consider three super records $\{R_1, R_2, R_3\}$ instead of $\{r_1, \ldots, r_6\}$. Since only the similarity of (R_1, R_2) is above 0.5, we merge them into a new super record R_1. (assume that 1=unoin$(1, 2)$) Finally, the stop condition is satisfied. r_1, r_2, r_4, r_6 are merged into super record R_1, thus they refer to the same entity. Similarly, r_3 and r_5 refer to the same entity.

To begin with, we show the correctness of *HERA* as follows.

Proposition 5. HERA *is optimal in terms of the number of comparisons between records in the worst case.*

Authorized licensed use limited to: Renmin University. Downloaded on July 18,2022 at 09:11:56 UTC from IEEE Xplore. Restrictions apply.

TABLE 1
Technical Characteristics of D_{m1}-D_{m4}

	D_{m1}	D_{m2}	D_{m3}	D_{m4}
n	100000	200000	300000	400000
# *of entity*	2986	4134	5104	6079
# *of distinct attribute*	27	31	36	32

TABLE 2
Parameters for Different Datasets

	D_{m1}	D_{m2}	D_{m3}	D_{m4}
$\lvert\mathcal{S}\rvert$ (Byte)	6.1×10^6	8.4×10^6	1.1×10^7	1.3×10^7
$\overline{m}$	72.1	89.4	76.3	98.5
k	46	51	39	49

Proof 2. PROPOSITION 5 For the ease of statement, we treat the *match* and *merge* process as black boxes in our proof. In *HERA*, we use array vis to ensure that each record pair are compared at most once in the worst case. If some record pair are not compared, their corresponding super records must be compared in some round. If there exists some algorithm, named A, it can compute the ER result using less number of comparisons. There must exist some pair of record, denoted by r_1 and r_2, so that both that record pair and their super record pair are not compared by A. We can construct the same match and merge process for record pairs unless (r_1, r_2). For r_1 and r_2, we construct match function to let them to be similar, i.e., $r_1 \approx r_2$. Then, they will be merged into a super record, named r_3, which belongs to the final result of ER. However, in algorithm A, r_3 will not be merged, and not appear in the solution, which is a contradiction. Thus our proof completes. □

Next, we analyze the time complexity of *HERA* as follows.

Proposition 6. *Time complexity of* HERA *is* $O(k(|\mathcal{V}| + |\mathbb{R}|(\overline{m}^3 + |\overline{\mathcal{V}}|log|\mathcal{V}|)))$.

k is the number of iterations, $|\mathbb{R}|$ is the number of candidate record pairs, $\overline{m}$ is the average number of points in the simplified bipartite graph, and $|\overline{\mathcal{V}}|$ is the average number of value pairs covered by any compared record pairs in index. We show the proof in [29].

7 EXPERIMENT

In this section, we report the experimental evaluation on comparing our method with the state-of-the-art and testing performance under different parameter settings.

7.1 Experimental Setup

Data Sets. We employed a real data set D_{movies} in [1], [11], which is a collection of movies shared among IMDB and DBPedia.[1] To derive the ground truth, we relied on the "*imdbid*" attribute in the profiles of the DBPedia movies. Since D_{movies} is pretty large, we extracted four data sets from D_{movies}, D_{m1}, D_{m2}, D_{m3} and D_{m4}, respectively. Typically, we randomly select four attributes combinations to generate the schemas of datasets from the overall distinct attribute list extracted from D_{movies}, and followed by extracting corresponding records randomly under given schema.

Table 1 lists the technical characteristics D_{m1}-D_{m4}. The number of entities could be counted by ground truth. Whether two attributes are distinct is distinguished manually, thus the number of distinct attributes can be obtained.

1. http://wiki.dbpedia.org/Datasets

Since most state-of-the-art focus on records under a predefined schema (as mentioned in related work), in order to make the experiment comparable, we conduct it as follows: we perform *schema matching* and *data exchange* on the above four datasets to generate another 8 datasets. Next we take D_{m1} as an example. We generated two datasets D_{m1-S} and D_{m1-L} from D_{m1} as follows.

1) *Schema matching*: Let A_1 be the number of distinct attributes contained in D_{m1}. Whether two attributes are matched can be distinguished manually. Since in most scenarios a target schema is user-defined and created for individual goals, we randomly selected part of distinct attributes from source schemas contained in D_{m1} to generate the target schema. To vary the information content between source schemas and target schema, we randomly chose $A_1/3$ and $2 * A_1/3$ distinct attributes in D_{m1} to generate the target schema of D_{m1-S} and D_{m1-L}, respectively. Then we decide the schema matchings manually.
2) *Data exchange*: Following [31], we transformed the schema matchings to corresponding tgds (*tuple generating dependency*) and used the data exchange modeling tool [32] to convert instances in D_{m1} to D_{m1-S} and D_{m1-L}.

The same operations are executed on D_{m2}, D_{m3}, D_{m4} to generate D_{m2-S}, D_{m2-L}, D_{m3-S}, D_{m3-L}, D_{m4-S}, D_{m4-L}, respectively. In the remainder of our paper, we call such 8 generated datasets as *homogeneous* datasets.

Implementation. We build index for D_{m1}-D_{m4} and leave the details in technical report [29].

Our experimental evaluations mainly consist of two aspects, *efficieny* and *effectiveness*.

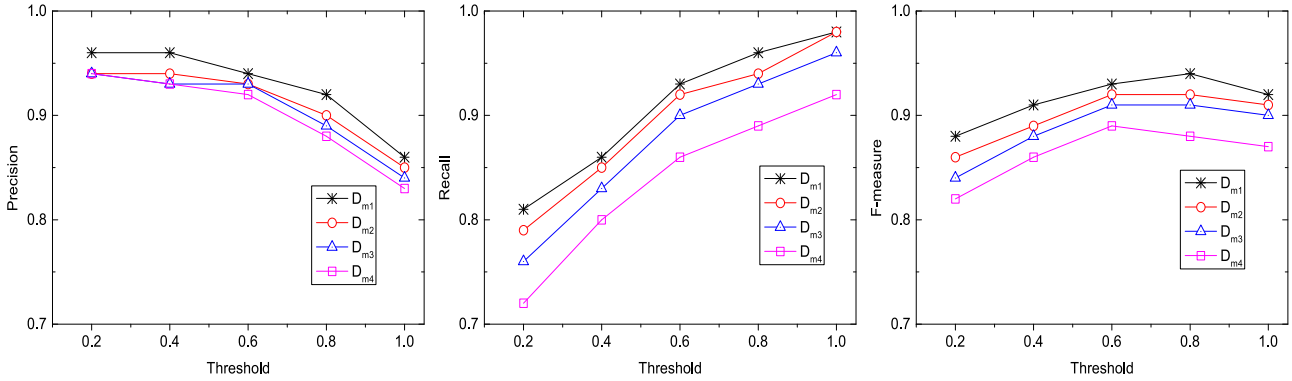
Regarding efficiency, we first examined some key parameters for D_{m1}-D_{m4}: $\mathcal{S}$, the size of index; $\overline{m}$, the average number of points in simplified bipartite graph described in Section 5.1; and k, the number of iterations. Then we reported run time and the number of comparisons of HERA conditioned on various thresholds on D_{m1}-D_{m4}.

Regarding effectiveness, we tested the quality of results returned by HERA on D_{m1}-D_{m4}. Next, we compared HERA with the other three state-of-art algorithms, R − Swoosh [3], CC [5] (Correlation clustering) and CR [4] (Collective ER) on 8 homogeneous datasets.

TABLE 3
Time Cost over Different Datasets with/without Index ($\delta = .8$)

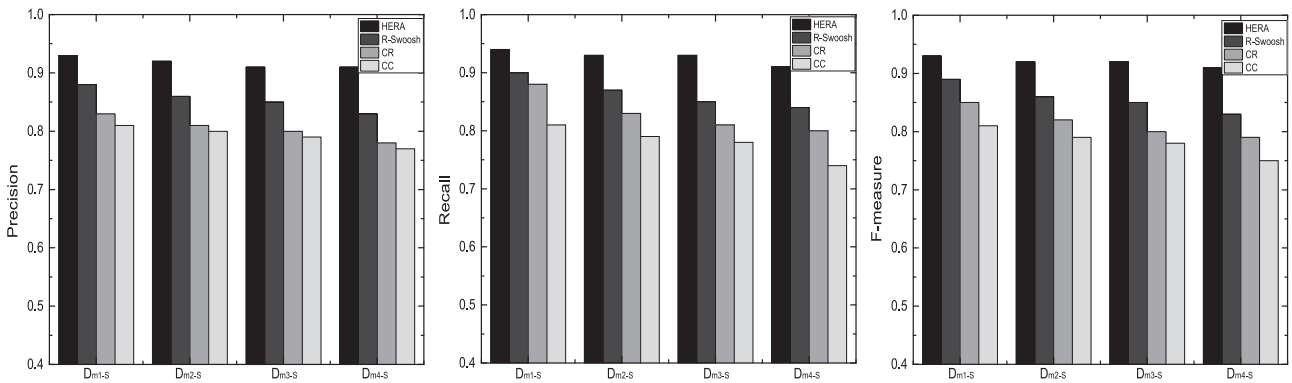
	D_{m1}	D_{m2}	D_{m3}	D_{m4}
without index (seconds)	56.12	103.44	171.26	210.93
with index (seconds)	7.13	7.41	7.52	8.83

Authorized licensed use limited to: Renmin University. Downloaded on July 18,2022 at 09:11:56 UTC from IEEE Xplore. Restrictions apply.



(a) Precision (b) Recall (c) F-measure

Fig. 9. Effectiveness test on D_{m1} to D_{m4}.



(a) Precision (b) Recall (c) F-measure

Fig. 10. Effectiveness test on D_{m1-S} to D_{m4-S}.

In this section, we report the quality of results on D_{m1-S} to D_{m4-S}. The results on the other four datasets are shown in technical report [29]. We implemented the algorithms in C++ on a Windows machine with Intel Core i5 processor.

Measure. We compared the results of our method with ground truth and measured its quality by *precision*, *recall* and *F1-measure*. *Presicion* is defined as the proportion of correctly identified record pairs to the record pairs generated by HERA. *Recall* is the proportion of correctly identified record pairs to the correct record pairs based on the ground-truth entities, and F1-measure is the harmonic average of precision and recall: $2/(1/precision + 1/recall)$.

7.2 Efficiency

We conducted three experiments to show the performance of HERA. Table 2 lists $|\mathcal{S}|$, $\overline{m}$ and k on four datasets. The average edges of simplified bipartite graph m is a fairly small number, indicating that the simplification approach reduces graph size significantly. In contrast, the number of value pair in index is a considerably large one, up to $1.3 * 10^7$. Fig. 11 depicts the execution time on different datasets. As performing HERA on datasets with larger size, it takes more time. When δ increases, time cost on different datasets varies more slightly. Typically, as δ is 0.8, HERA finished in around 8 seconds for all datasets. Next we test how the index affect the overall time cost. From Table 3, our method with index outperforms the one without index greatly, indicating that leveraging index reduces the number of candidate pairs very effectively.

7.3 Effectiveness

In this part, we first conducted experiments to show the quality of results on four heterogeneous datasets. Next we compared with some state-of-art approaches, R − Soosh, CR and CC on datasets D_{m1-S}, D_{m2-S}, D_{m3-S} and D_{m4-S}.

To examine how threshold affects precision, we report the comparison results in Fig. 9a. Generally, we observe a slight decline for precision as the size of datasets expands. In contrast, the decline turns pronounced when δ drops. It decreases by 9 percent on average from D_{m1} to D_{m4}. In Fig. 9b, we plots recall w.r.t. different thresholds. The curves climb dramatically as δ increases. On D_{m1}, recall is 0.98 when $\delta = 1$ and it turns 0.81 when $\delta = 0.2$. Regarding the size of datasets, similarly, the larger size, the lower recall. On D_{m4}, the worst recall is 0.72 ($\delta = 0.2$) while the worst recall on D_{m1} turns 0.81. F1-measure is presented in Fig. 9c: first, F-measure increases till the best value and then drops slightly. The best value for D_{m1} is 0.8, and for others is roughly 0.6. Second, as we queried more records, F-measure drops slightly. The average F-measure on D_{m1} is 91.6, on D_{m4} is 86.4, decreasing by 4.4 percent.

Next we compared the accuracy of HERA with R − Soosh, CR and CC in terms of precision, recall and F-measure. Fig. 10a shows the precision. Compared with three competitors, HERA has the best performance. Specifically, the average precision exceeds 0.9, which improves R − SWoosh by 6 percent, CR by 12 percent and CC by 13 percent. Among them, CR and CC have similar precision on four datasets. Considering recall shown in Fig. 10b, HERA also outperforms others on all datasets. The average

Authorized licensed use limited to: Renmin University. Downloaded on July 18,2022 at 09:11:56 UTC from IEEE Xplore. Restrictions apply.

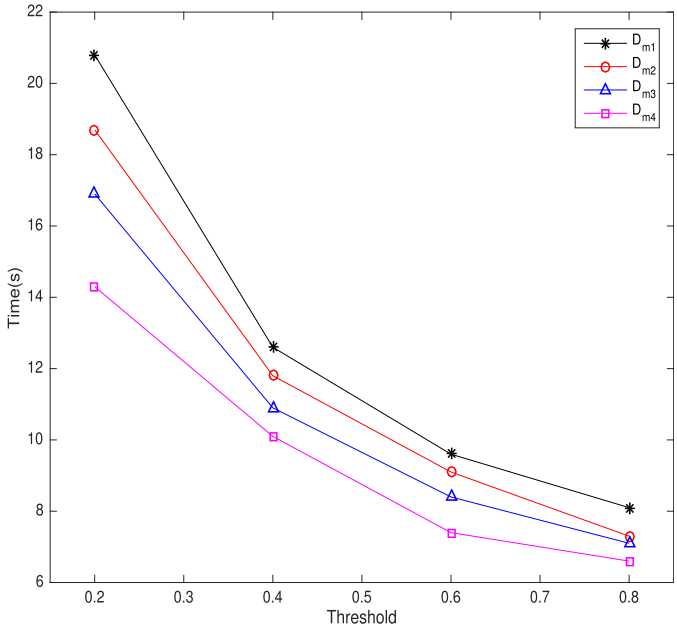


Fig. 11. Runtime.

recall of HERA is 92.75, beats R − Swoosh by 6 percent, CR by 10 percent and CC by 16 percent. CR and CC are slightly sensitive to the size of datasets and CC obtains a low recall below 0.8 on four datasets. On datasets with relatively small size, R − Swoosh has competitive performance. Its recall rate reaches nearly 0.9. Finally, regarding F-measure, HERA obtains a significant improvement in comparison to the other methods. On average, it outperforms R − Swoosh by 6 percent, CR by 11 percent, CC by 15 percent. Furthermore, HERA is insensitive conditioned on datasets with different sizes (range roughly from 10,000 to 40,000) while the F-measure of other methods shows a pronounced decline. The decline of quality for the other three methods shows that the difference of information content between target schema and source schemas (In Section 7.1, we set $\frac{\#\text{ of attributes in } D_{m1-S}}{\#\text{ of attributes in } D_{m1}} = \frac{1}{3}$, and the ratio also holds for the other three datasets) lowers the quality of ER on homogeneous records set significantly. In contrast, HERA can collect sufficient information in heterogeneous sources to improve ER on datasets with homogeneous schema.

The reason why HERA outperforms others is information loss between heterogeneous and homogeneous schema. The three comparable technique could only tackle ER under homogeneous schema, where information loss is inevitable, thus lowering quality of results. While HERA implements ER under heterogeneous schema, and have full information. The solution to *description difference* enables a high quality of final results.

8 Conclusion

In this paper, to solve ER under heterogeneous environment, we propose HERA. Basically, we describe a *merge-and-compare* mechanism to solve *description difference*. Regarding efficiency, we design an index to generate candidates and speed up record similarity computation. Without the information of schema matchings, we present instance-based and schema-based method to verify the candidates. We show that two methods can benefit each other to accelerate similarity computation and improve the accuracy. Experimental results show that our methods can significantly improve the state-of-art even on records under a predefined schema.

Acknowledgments

This paper was partially supported by NSFC grant U1509216, U1866602,61602129.

References

[1] G. Papadakis, E. Ioannou, C. Niederée, and P. Fankhauser, "Efficient entity resolution for large heterogeneous information spaces," in *Proc. ACM Int. Conf. Web Search Data Mining*, 2011, pp. 535–544.
[2] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios, "Duplicate record detection: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 1, pp. 1–16, Jan. 2007.
[3] O. Benjelloun, H. Garcia-Molina, D. Menestrina, Q. Su, S. E. Whang, and J. Widom, "Swoosh: A generic approach to entity resolution," *VLDB J.*, vol. 18, pp. 255–276, 2009.
[4] I. Bhattacharya and L. Getoor, "Collective entity resolution in relational data," *ACM Trans. Knowl. Discovery Data*, vol. 1, no. 1, 2007, Art. no. 5.
[5] N. Ailon, M. Charikar, and A. Newman, "Aggregating inconsistent information: Ranking and clustering," *J. ACM*, vol. 55, 2008, Art. no. 23.
[6] S. N. Minton, C. Nanjo, C. A. Knoblock, M. Michalowski, and M. Michelson, "A heterogeneous field matching method for record linkage," in *Proc. IEEE Int. Conf. Data Mining*, 2005, pp. 314–321.
[7] S. Rong, X. Niu, E. W. Xiang, H. Wang, Q. Yang, and Y. Yu, "A machine learning approach for instance matching based on similarity metrics," in *Proc. Int. Semantic Web Conf.*, 2012, pp. 460–475.
[8] J. Wang, T. Kraska, M. J. Franklin, and J. Feng, "CrowdER: Crowdsourcing entity resolution," *Proc. VLDB Endowment*, vol. 5, pp. 1483–1494, 2012.
[9] A. Marcus, E. Wu, D. Karger, S. Madden, and R. Miller, "Human-powered sorts and joins," *Proc. VLDB Endowment*, vol. 5, pp. 13–24, 2011.
[10] G. Papadakis, E. Ioannou, T. Palpanas, C. Niederee, and W. Nejdl, "A blocking framework for entity resolution in highly heterogeneous information spaces," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 12, pp. 2665–2682, Dec. 2013.
[11] G. Papadakis, E. Ioannou, C. Niederée, T. Palpanas, and W. Nejdl, "To compare or not to compare: Making entity resolution more efficient," in *Proc. Int. Workshop Semantic Web Inf. Manage.*, 2011, Art. no. 3.
[12] V. Efthymiou, K. Stefanidis, and V. Christophides, "Big data entity resolution: From highly to somehow similar entity descriptions in the web," in *Proc. IEEE Int. Conf. Big Data*, 2015, pp. 401–410.
[13] A. Doan and A. Y. Halevy, "Semantic integration research in the database community: A brief survey," *AI Mag.*, vol. 26, pp. 83–94, 2005.
[14] N. Koudas, S. Sarawagi, and D. Srivastava, "Record linkage: Similarity measures and algorithms," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2006, pp. 802–803.
[15] X. L. Dong and D. Srivastava, "Big data integration," in *Proc. IEEE 29th Int. Conf. Data Eng.*, 2013, pp. 1245–1248.
[16] V. Christophides, V. Efthymiou, and K. Stefanidis, "Entity resolution in the web of data," *Synthesis Lectures Semantic Web*, vol. 5, no. 3, pp. 1–122, 2015.
[17] M. Kejriwal and D. P. Miranker, "An unsupervised instance matcher for schema-free RDF data," *Web Semantics: Sci. Serv. Agents World Wide Web*, vol. 35, pp. 102–123, 2015.
[18] Y. Zheng, J. Wang, G. Li, R. Cheng, and J. Feng, "QASCA: A quality-aware task assignment system for crowdsourcing applications," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2015, pp. 1031–1046.
[19] G. Li, J. Wang, Y. Zheng, and M. J. Franklin, "Crowdsourced data management: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 9, pp. 2296–2319, Sep. 2016.
[20] G. Li, C. Chai, J. Fan, X. Weng, J. Li, Y. Zheng, Y. Li, X. Yu, X. Zhang, and H. Yuan, "CDB: Optimizing queries with crowd-based selections and joins," in *Proc. ACM Int. Conf. Manage. Data*, 2017, pp. 1463–1478.
[21] G. Li, Y. Zheng, J. Fan, J. Wang, and R. Cheng, "Crowdsourced data management: Overview and challenges," in *Proc. ACM Int. Conf. Manage. Data*, 2017, pp. 1711–1716.

Authorized licensed use limited to: Renmin University. Downloaded on July 18,2022 at 09:11:56 UTC from IEEE Xplore. Restrictions apply.

[22] G. Li, C. Chai, J. Fan, X. Weng, J. Li, Y. Zheng, Y. Li, X. Yu, X. Zhang, and H. Yuan, "CDB: A crowd-powered database system," *Proc. VLDB Endowment*, vol. 11, no. 12, pp. 1926–1929, 2018.
[23] S. Lacoste-Julien, K. Palla, A. Davies, G. Kasneci, T. Graepel, and Z. Ghahramani, "SiGMa: Simple greedy matching for aligning large knowledge bases," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2013, pp. 572–580.
[24] C. Böhm, G. de Melo, F. Naumann, and G. Weikum, "LINDA: Distributed web-of-data-scale entity matching," in *Proc. 21st ACM Int. Conf. Inf. Knowl. Manage.*, 2012, pp. 2104–2108.
[25] C. Shao, L.-M. Hu, J.-Z. Li, Z.-C. Wang, T. Chung, and J.-B. Xia, "RiMOM-IM: A novel iterative framework for instance matching," *J. Comput. Sci. Technol.*, vol. 31, no. 1, pp. 185–197, 2016.
[26] D. B. West, et al., *Introduction to Graph Theory*. Upper Saddle River, NJ, USA: Prentice Hall, 2001.
[27] L. Getoor and A. Machanavajjhala, "Entity resolution: Tutorial," University of Mariland, 2012. [Online]. Available: http://www.cs.umd.edu/getoor/Tutorials/ER_VLDB2012. pdf
[28] W. Wang, "Similarity join algorithms: An introduction," *SEBD*, vol. 8, p. 2, 2008.
[29] Y. Lin, H. Wang, J. Li, and H. Gao, "Efficient entity resolution on heterogeneous records," Harbin Institute of Technology, Harbin, China, 2017.
[30] T. H. Cormen, *Introduction to Algorithms*. Cambridge, MA, USA: MIT Press, 2009.
[31] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa, "Data exchange: Semantics and query answering," *Theoretical Comput. Sci.*, vol. 336, no. 1, pp. 89–124, 2005.
[32] R. Pichler and V. Savenkov, "DEMo: Data exchange modeling tool," *Proc. VLDB Endowment*, vol. 2, pp. 1606–1609, 2009.

Yiming Lin received the master's degree from the Harbin Institute of Technology. His research interests include data quality and data integration.

Hongzhi Wang is a professor and doctoral supervisor with the Harbin Institute of Technology. His research area is data management, including data quality, XML data management, and graph management. He has published more than 100 papers in refereed journals and conferences. He is a recipient of the outstanding dissertation award of CCF, Microsoft Fellow, and IBM PhD Fellowship.

Jianzhong Li is a professor and doctoral supervisor with the Harbin Institute of Technology. He is a senior member of CCF. His research interests include database, parallel computing, and wireless sensor networks, etc.

Hong Gao is a professor and doctoral supervisor with the Harbin Institute of Technology. She is a senior member of CCF. Her research interests include data management, wireless sensor networks, and graph database, etc.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/csdl.

Authorized licensed use limited to: Renmin University. Downloaded on July 18,2022 at 09:11:56 UTC from IEEE Xplore. Restrictions apply.