Rule-Based Method for Entity Resolution

Lingli Li, Jianzhong Li, and Hong Gao

Abstract—The objective of entity resolution (ER) is to identify records referring to the same real-world entity. Traditional ER approaches identify records based on pairwise similarity comparisons, which assumes that records referring to the same entity are more similar to each other than otherwise. However, this assumption does not always hold in practice and similarity comparisons do not work well when such assumption breaks. We propose a new class of rules which could describe the complex matching conditions between records and entities. Based on this class of rules, we present the rule-based entity resolution problem and develop an on-line approach for ER. In this framework, by applying rules to each record, we identify which entity the record refers to. Additionally, we propose an effective and efficient rule discovery algorithm. We experimentally evaluated our rule-based ER algorithm on real data sets. The experimental results show that both our rule discovery algorithm and rule-based ER algorithm can achieve high performance.

Index Terms—Entity resolution, rule learning, data cleaning

1 INTRODUCTION

In many applications, a real-world entity may appear in multiple data sources so that the entity may have quite different descriptions. For example, there are several ways to represent a person’s name or a mailing address. Thus, it is necessary to identify the records referring to the same real-world entity, which is called Entity Resolution (ER). ER is one of the most important problems in data cleaning and arises in many applications such as information integration and information retrieval. Because of its importance, it has attracted much attention in the literature [27].

Traditional ER approaches obtain a result based on similarity comparison among records, assuming that records referring to the same entity are more similar to each other (compact set property [17]). However, such property may not hold so traditional ER approaches cannot identify records correctly in some cases. We use the following example to illustrate one of the cases.

Example 1. Table 1 shows seven authors with name “wei wang” identified by $o_1$. By accessing to the authors home pages containing their publications, we manually divide the seven authors into three clusters. The records with IDs $o_{11}$, $o_{12}$, and $o_{13}$ refer to the person in UNC, denoted as $e_1$, the records with IDs $o_{21}$ and $o_{22}$ refer to the person in UNSW, denoted as $e_2$, and the records with IDs $o_{31}$ and $o_{32}$ refer to the person in Fudan University, denoted as $e_3$. The task of entity resolution is to identify $e_1$, $e_2$ and $e_3$ using the information in Table 1. Since all these records have identical name but different set of coauthors in different papers, the similarity between any two records $X$ and $Y$, denoted by $Sim(X,Y)$, is determined by the similarity of coauthors. To measure the similarity between sets, Jaccard similarity [3] is often used, and thus the similarity between any two records, $X$ and $Y$, is defined as $Sim(X,Y) = \frac{|\text{coauthors}[X] \cap \text{coauthors}[Y]|}{|\text{coauthors}[X] \cup \text{coauthors}[Y]|}$.

Thus, we have the following facts:

- $Sim(o_{11}, o_{12}) < Sim(o_{11}, o_{31})$ since $Sim(o_{11}, o_{12}) = 0$ and $Sim(o_{11}, o_{31}) = \frac{3}{4}$, and
- $Sim(o_{12}, o_{13}) < Sim(o_{12}, o_{21})$ since $Sim(o_{12}, o_{13}) = \frac{1}{3}$ and $Sim(o_{12}, o_{21}) = \frac{1}{2}$.

The result shows that the similarity between $o_{11}$ and $o_{12}$ is smaller than the similarity between $o_{11}$ and $o_{31}$ even though $o_{11}$ and $o_{12}$ refer to the same entity while $o_{11}$ and $o_{31}$ refer to different entities. It is obvious that we are unable to get the correct ER result of the example by applying similarity comparison between records. Similar to Jaccard, other similarity functions, such as cosine similarity and TF-IDF, also have the same problem. As similarity comparisons can not be applied in this case, we have the following observations.

Observation 1. The existence of some attribute-value pairs are useful to identify records.

Take Table 1 as an example. The attribute-value pair (coauthors,”lin”) occurs only in the records referring to $e_2$. Thus, the existence of (coauthors,”lin”) can be used to identify records referring to $e_2$. Similarly, the existence of (coauthors,”kum”) and (coauthors,”shi”) can be used to identify records referring to $e_1$ and $e_3$ respectively.

Observation 2. The nonexistence of some attribute-value pairs are also useful to identify records.

Taking Table 1 as an example again, the coauthors of the record $o_{11}$ includes only “zhang”. Since “zhang” occurs in both $o_{11}$ and $o_{31}$, the existence of (coauthors,”zhang”) can distinguish the records referring to $e_1$ or $e_3$ from the other records, but cannot distinguish the records referring to $e_1$ and $e_3$. However, the nonexistence of (coauthors,”shi”) can be used to rule out the possibility of $o_{11}$ referring to $e_3$ since the existence of (coauthors,”shi”) can identify all the records referring to $e_3$. Thus, the existence of (coauthors,”zhang”) and the nonexistence of (coauthors,”shi”) can be used together to identify the records referring to $e_1$. 
Based on the observations, we are able to develop the following rules to identify records in Table 1.

- $R_1$: \( \forall o_i, \text{if } o_i[\text{name}] \text{ is } \text{“wei wang”} \text{ and } o_i[\text{coauthors}] \text{ includes } \text{“kum”, then } o_i \text{ refers to entity } e_1; \)
- $R_2$: \( \forall o_i, \text{if } o_i[\text{name}] \text{ is } \text{“wei wang”} \text{ and } o_i[\text{coauthors}] \text{ includes } \text{“lin”, then } o_i \text{ refers to entity } e_2; \)
- $R_3$: \( \forall o_i, \text{if } o_i[\text{name}] \text{ is } \text{“wei wang”} \text{ and } o_i[\text{coauthors}] \text{ includes } \text{“shi”, then } o_i \text{ refers to entity } e_3; \)
- $R_4$: \( \forall o_i, \text{if } o_i[\text{name}] \text{ is } \text{“wei wang”} \text{ and } o_i[\text{coauthors}] \text{ includes } \text{“zhang”} \text{ and excludes } \text{“shi”, then } o_i \text{ refers to entity } e_4. \)

This example shows that the disadvantages of the traditional ER methods can be overcome by employing rules generated from the entities’ information. This findings motivate us to develop a rule-based entity resolution method. This gives rise to the following challenges.

1) How to define the rules as described in Example 1 and how to define the properties to be satisfied by the rules to ensure a good performance of entity resolution both in efficiency and effectiveness?

2) How to discover the rules from a given training data set to support ER efficiently and effectively?

Note that, to discover rules to resolve entities, the information of real-world entities are required. In practice, such information can be collected from many different sources. For example, the information of clients can be obtained from the master data maintained by companies, the information, such as skills, education, etc., of professional persons can be collected from LinkedIn, and the information of researchers and scientists can be collected from ResearchGate. In this paper, we assume that the information of entities are already collected.

3) How to use the rules to identify records efficiently and effectively?

4) How to maintain the rules when entity information is changed?

This paper aims at the aforementioned problems, and the main contributions of the paper are as following.

1) The syntax and semantics of the rules for ER are designed, and the independence, consistency, completeness and validity of the rules are defined and analyzed.

2) An efficient rule discovery algorithm based on training data is proposed and analyzed.

3) An efficient rule-based algorithm for solving entity resolution problem is proposed and analyzed.

4) A rule maintaining method is proposed when entity information is changed.

5) Experiments are performed on real data to verify the effectiveness and efficiency of the proposed algorithms.

In fact, our method and traditional ER approaches can be considered as the complementary to each other and be applied together. This is because our rule-based method can identify records which cannot be resolved by traditional ER methods and traditional ER methods can identify most of the records effectively and do not require the availability of correct entity set. In this way, the limitations of both methods can be overcome.

### 2 RULES FOR ENTITY RESOLUTION

In this section, a rule system for entity resolution, called ER-rule, is defined. We can see that each rule in Example 1 consists of two clauses. (1) The If clause includes constraints on attributes of records, such as “including zhong in coauthors”, and (2) the Then clause indicates the real world entity referred by the records that satisfy the first clause of the rule, such as “refers to entity $e_i$.” Thus, we use $A \Rightarrow B$ to express the rules “$\forall o_i, \text{if } o_i$ satisfies $A$ Then $o_i$ refers to $B$” for ER. We denote the left-hand side and the right-hand side of a rule as LHS($r$) and RHS($r$) respectively.

#### 2.1 Syntax

An ER-rule is syntactically defined as $T_1 \land \cdots \land T_m \Rightarrow e$, where $T_i (1 \leq i \leq m)$ is a clause with the form of $(A_i, op_i, v_i), (v_i, op, A_i), \neg(A_i, op_i, v_i)$ or $\neg(v_i, op, A_i)$, where $A_i$ is an attribute, $v_i$ is a constant in the domain of $A_i$, and $op_i$ can be any domain-independent operator defined by users, such as exact match operator $=$, fuzzy match operator $\approx$ [16] for string value, $\leq$ for numeric value, or $\in$ for set value. The clause with form $(A_i, op_i, v_i)$ or $(v_i, op, A_i)$ is called positive clause, and the clause with form $\neg(A_i, op_i, v_i)$ or $\neg(v_i, op, A_i)$ is called negative clause.

Each ER-rule $r$ can be assigned a weight $w(r)$ in $[0, 1]$ to reflect the level of confidence that $r$ is correct. Intuitively, the more records are identified by an ER-rule $r$, the more possible $r$ is correct. Therefore, given a data set $S$, we define the weight of each ER-rule $r$ as:

$$w(r) = \frac{|S(r)|}{|S(RHS(r))|},$$

where $S(r)$ denotes the records in $S$ that are identified by $r$ and $S(RHS(r))$ denotes the records in $S$ that refer to entity RHS($r$).

In the rest of the paper, we assume that the operator $op_i$ for each attribute $A_i$ is given. In this way, a positive clause $(A_i, op_i, v_i)$ can be abbreviated as an attribute-value pair $(A_i, v_i)$. Similarly, a negative clause $\neg(A_i, op_i, v_i)$ can be abbreviated as $\neg(A_i, v_i)$.

**Example 2.** The rules given in Example 1 can be expressed as the following ER-rules respectively. For simplicity we write coa rather than coauthors.

$$r_1: \text{name} = \text{“wei wang”} \land (\text{“kum”} \in \text{coa} ) \Rightarrow e_1,$$

$$r_2: \text{name} = \text{“wei wang”} \land (\text{“lin”} \in \text{coa} ) \Rightarrow e_2,$$

<table>
<thead>
<tr>
<th>Table 1: Paper-Author Records</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>e1</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>e2</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>e3</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
where $x \in Y$ represents that attribute $Y$ includes value $x$. For $r_4$, (name = “wei wang”), (“zhang” $\in$ coa) are positive clauses and $¬$(“shi” $\in$ coa) is a negative clause, where name, coa are attributes, $=\in$ are operators and “wei wang”, “zhang”, “shi” are values. $r_4$ can also be simplified as (name,“wei wang”) $\land$ (“coa,”“zhang”) $\land$ $¬$(coa,“shi”) $\Rightarrow$ $e_1$.

### 2.2 Semantics

In the following definitions, we let $o$ be a record, $S$ be a data set, $r$ be an ER-rule and $R$ be an ER-rule set. For the convenience of discussion, we assume the mapping from each record in $S$ to its actual entity is given.

Since an ER-rule does not include disjunct clauses, we define the condition of matching the left-hand and right-hand sides of rule as follows.

**Definition 1.** $o$ matches the LHS of $r$ if $o$ satisfies all the clauses in LHS($r$). $o$ matches the RHS of $r$ if $o$ refers to entity RHS($r$).

**Definition 2.** $o$ satisfies $r$, denoted by $o \vdash r$, if $o$ does not match LHS($r$) or matches RHS($r$).

In another word, record $o$ does not satisfy ER-rule $r$ if and only if $o$ matches LHS($r$) and does not match RHS($r$). This definition is based on that “if $A$ then $B$” is equivalent to “$¬A \lor B$” in first order logic system.

**Example 3.** Consider records in Table 1 and ER-rule $r_1$ in Example 2. For $o_{12}$ and $o_{13}$, they satisfy $r_1$ because they both match LHS($r_1$) and RHS($r_1$). For the other records in Table 1, they also satisfy $r_1$ because none of them matches LHS($r_1$). Therefore $r_1$ is satisfied by all the records in Table 1.

**Definition 1 and 2** presents the semantics of an ER-rule, that is, if a record $o$ makes LHS($r$) true then $o$ must refer to the entity denoted by RHS($r$).

Moreover, Definition 2 ($\vdash$) can be extended as follows.

- $S \vdash r$ if $o \vdash r$ for $\forall o \in S$, and we say $r$ is a valid rule for data set $S$ if $S \vdash r$.
- $S \vdash r$ if $o \vdash r$ for $\forall r \in R$.

Based on the above definitions, we say $R$ entails $r$, denoted by $R \models r$, if $o \vdash r$ for any possible record $o$.

**Definition 3.** $o$ is identified by $r$, if $o$ matches both LHS($r$) and RHS($r$). Note that, if $o$ is identified by $r$, $o$ must satisfy $r$. If $o$ satisfies $r$, $o$ might not be identified by $r$.

**Example 4.** Consider $o_{11}$ in Table 1 and the ER-rule set $R = \{r_1, r_2, r_3, r_4\}$ in Example 2. $o_{11}$ matches LHS($r_4$) because $o_{11}$[name] = “wei wang”, “zhang” $\in$ [coa] and “shi” $\not\in$ $o_{11}$[coa]. $o_{11}$ $\vdash r_1$ and $o_{11}$ is identified by $r_1$ because $o_{11}$ matches LHS($r_4$) and RHS($r_4$). $o_{11}$ $\vdash r_1$ but $o_{11}$ is not identified by $r_1$ because $o_{11}$ does not match LHS ($r_1$). $o_{11}$ $\vdash r$ because $o_{11}$ $\vdash r$ for $\forall r \in R$.

### 2.3 Properties of ER-Rule Set

Given an ER-rule set $R$ and a data set $S$, to ensure $R$ performs well on $S$, we require that (1) there is no false matches between record and entity (validity); (2) there is no conflicting decisions by $R$ (consistency); (3) each record in $S$ can be mapped to an entity by $R$ (completeness) and (4) there is no redundant rules in $R$ (independence). Now we present the formal definitions of these properties.

**Definition 4 (Validity).** $R$ is valid for $S$ if each ER-rule in $R$ is valid for $S$.

**Definition 5 (Consistency).** $R$ is consistent for $S$ if $o$ matches both LHS($r_1$) and RHS($r_2$) then RHS($r_1$) = RHS($r_2$) for $\forall o \in S$ and $\forall r_1, r_2 \in R$.

**Definition 6 (Completeness).** $R$ is complete for $S$ if for $\forall o \in S$, $\exists r \in R$ such that $o$ matches RHS($r$).

**Definition 7 (Independence).** $R$ is independent if each ER-rule $r$ in $R$ satisfies that $R \setminus \{r\}$ does not entail $r$, denoted as $R \setminus \{r\} \not\models r$.

The following proposition shows that an ER-rule set will contain redundant rules if it is not independent.

**Proposition 1.** If both $r$ and $R$ are valid for $S$ and $R \models r$, then for any record $o$ in $S$, if $o$ is identified by $r$, $o$ must be identified by $R$.

**Proof.** We prove that if there exists one record $o$ in $S$ such that $o$ is identified by $r$ but $o$ is not identified by $R$, then there exists no rule $r'$ in $R$ such that $o$ matches RHS($r'$). Otherwise, let $o$ be a record which is not identified by $R$ and there exists one rule $r'$ in $R$ such that $o$ matches RHS($r'$). Since $o$ is not identified by $R$, $o$ does not match RHS($r'$). Then $o \not\models r'$. Thus $R$ is not valid for $S$, which contradicts to our assumption. Therefore the conclusion that if there exists one record $o$ in $S$ such that $o$ is identified by $r$ but $o$ is not identified by $R$, then there exists no rule $r'$ in $R$ such that $o$ matches RHS($r'$) is true.

Now we prove that if there exists one record $o$ in $S$ such that $o$ is identified by $r$ but $o$ is not identified by $R$, then there exists a record $o'$ such that $o' \vdash r$ but $o' \not\vdash r$. We construct the record $o'$ as follows: let $o' = o$ and $o'$ does not refer to RHS($r$). Since $o' = o$ and there exists no rule $r'$ in $R$ such that $o$ matches RHS($r'$), $o' \vdash r$. Thus $R \not\models r$ because $o' \vdash R$ and $o' \not\vdash r$. This is in contradiction with our assumption that $R \models r$. Therefore, the clause is true.

**Proposition 2.** If $R$ is valid for $S$, $R$ is consistent for $S$.

**Proof.** Suppose $R$ is valid but not consistent for $S$. There must exist a record $o \in S$ and ER-rules $r_1, r_2 \in R$ such that $o$ matches LHS($r_1$) and LHS($r_2$), and RHS($r_1$) $\neq$ RHS($r_2$). Since $o$ matches both LHS($r_1$) and LHS($r_2$) and $R$ is valid, $o$ should match both RHS($r_1$) and RHS($r_2$). Therefore, RHS($r_1$) = RHS($r_2$), which is in contradiction with the assumption that $R$ is valid but not consistent for $S$. Thus, the proposition is true.

Theorem 1 shows the expressive power of ER-rules.

**Theorem 1.** Given a data set $S = \{o_1, \ldots, o_n\}$, there is at least one ER-rule set $R$ that is independent, consistent, valid and complete for $S$.

**Proof.** For each record $o_i$ in $S$, let $e_i$ denote the entity to which $o_i$ refers, $U$ denote the set of attribute-value pairs that occur in records in $S$. We construct an ER-rule set $R = \{r_1, \ldots, r_n\}$ as follows. For each rule $r_i$ in $R$, RHS($r_i$) = $e_i$, and for $\forall (A_j, v_j) \in U$, if $(A_j, v_j) \in o_i$ then
(v_j \in A_j) \in \text{LHS}(r_j), \text{otherwise } (v_j \in A_j) \in \text{LHS}(r_j).

Note that, the equality operator(=) can be considered as a special case of operator \(\approx\) and \(\in\).

Hence, \(R\) has the following properties. For \(1 \leq i, j \leq n\),

1) if \(i = j, o_i\) matches \(\text{LHS}(r_j)\) and \(\text{RHS}(r_j)\);  
2) if \(i \neq j, o_i\) does not match \(\text{LHS}(r_j)\).

According to property (1), \(R\) is complete. According to the properties (1) and (2), \(R\) is valid. According to the Proposition 2, \(R\) is consistent.

To prove \(R\) is independent, we prove that \(R - \{r_i\} \neq r_i\) for each ER-rule \(r_i \in R\). For each ER-rule \(r_i \in R\), we construct a record \(o'_i\) as follows: \(o'_i = o_i\) but \(o'_i\) does not refer to \(\text{RHS}(r_i)\). Since \(o'_i\) matches \(\text{LHS}(r_i)\) and \(o'_i\) does not match \(\text{RHS}(r_i)\), then \(o'_i \neq r_i\). Moreover, since \(o'_i\) does not match \(\text{LHS}(r_i)\) for any ER-rule \(r_j \in R - \{r_i\}\), then \(o'_i \neq r_i\). Therefore \(R - \{r_i\} \neq r_i\) for each ER-rule \(r_i \in R\). Hence \(R\) is independent.

Therefore, given a data set \(S\), there exists at least one rule set \(R\) that is independent, consistent, valid and complete for \(S\).

3 Rule Discovery

Since it might be too expensive to construct ER-rules manually, we discuss how to discover useful rules from a training data set for efficient and effective entity resolution in this section. We assume that the operator on each attribute can be any domain-dependent operator defined by users.

First, we discuss the requirements of the discovered rule sets and present our framework of rule discovery (Section 3.1). Then we describe the algorithms in the rule discovery framework (Sections 3.2 and 3.3) and study the correctness and complexity of our algorithm (Section 3.4).

For exposition, proofs have been deferred to the Appendix.

For the convenience of discussion, some concepts are introduced first.

We classify ER-rules into two categories according to whether negative clauses are included.

Definition 8 (PR). PR is an ER-rule which only includes positive clauses.

Definition 9 (NR). NR is an ER-rule which includes at least one negative clause.

For example, \(r_1, r_2\) and \(r_3\) in Example 2 are all PRs while \(r_4\) is an NR.

Definition 10 (coverage). The coverage of clause \(T\) on \(S\), denoted as \(\text{Cov}_S(T)\), is the subset of \(S\) such that \(\text{Cov}_S(T) = \{o | o \in S, o \text{ satisfies } T\}\).

Accordingly, the coverage of rule \(r\) on \(S\), denoted as \(\text{Cov}_S(r)\), is the intersection of the coverage of each clause in \(r\), such that \(\text{Cov}_S(r) = \text{Cov}_S(T_1 \cap \text{Cov}_S(T_2) \cap \cdots \cap \text{Cov}_S(T_k)\), where \(\text{LHS}(r) = T_1 \land T_2 \land \cdots \land T_k\). Clearly, \(\text{Cov}_S(r) = \{o | o \in S, o \text{ matches } \text{LHS}(r)\}\).

The coverage of rule set \(R\) on \(S\), denoted as \(\text{Cov}_S(R)\), is the union of the coverage of each rule \(r \in R\) on \(S\), such that \(\text{Cov}_S(R) = \bigcup_{r \in R} \text{Cov}_S(r)\).

Note that, when \(S\) is clear from the context, \(\text{Cov}_S()\) is simplified as \(\text{Cov}()\).

Example 5. Let \(T = \{\text{"zhang"} \in \text{coa}\}\), \(S\) be the set of records in Table 1, we have \(\text{Cov}_S(T) = \{o_{11}, o_{31}\}\). Consider rules in Example 2, we have \(\text{Cov}_{r_1} = \{o_{12}, o_{13}\}, \text{Cov}_{r_2} = \{o_{21}, o_{22}\}, \text{Cov}_{r_3} = \{o_{41}, o_{42}\}, \text{Cov}_{r_4} = \{o_{11}\}\) and \(\text{Cov}_S(R) = S\) where \(R = \{r_1, r_2, r_3, r_4\}\).

Proposition 3. If \(\text{RHS}(r) = e_j\), then \(r\) is valid for \(S\) if each record in \(\text{Cov}_S(r)\) refers to \(e_j\).

Clearly, if rule \(r\) is valid for data set \(S\), \(\text{Cov}_S(r)\) is the set of records which are identified by \(r\). Accordingly, if rule set \(R\) is valid for \(S\), \(\text{Cov}_S(R)\) is the set of records in \(S\) that are identified by rules in \(R\).

3.1 Rule Discovery Problem

In this section, we define the problem of rule discovery. Let \(S = \{S_1, \ldots, S_m\}\) be the training data of data set \(S\), where each \(S_i\) in \(S\) is a subset of \(S\) which refer to the same entity, denoted by \(e_j\). As discussed, the discovered rule set should be independent, valid and complete for \(S\) to ensure a good performance of ER on \(S\). Thus these three properties should be taken into consideration for the rule discovery problem.

3.1.1 Requirements

Even though these properties are satisfied on the training data set, it cannot be ensured that the generated rule set can also perform well on the other data sets. To make the rule set suitable for ER for many data sets other than only suitable for the training data set, we require the discovered ER-rule set, denoted by \(R\), should also satisfy two requirements described as follows.

- **Length Requirement:** Given a threshold \(l\), each rule \(r\) in \(R\) satisfies that \(|r| \leq l\).

  To determine whether record \(o\) matches the LHS of ER-rule \(r\), we should check whether \(o\) satisfies each clause in \(\text{LHS}(r)\). Thus to guarantee the efficiency of rule-based ER (R-ER) and avoid overfitting, the length of each rule (the number of clauses) should be no more than a threshold.

- **PR Requirement:** each rule \(r\) in \(R\) is a PR.

  The reason why we give priority to PRs is that, positive literals lead to bounded spaces while negative literals lead to unbounded spaces. Therefore the discovered PRs are more possible to identify other data sets effectively than the discovered NRs.

However, both the length requirement and the PR requirement can decrease the expression power of ER-rules, so that there might be some records in \(S\) that cannot be identified by any valid PR with length no more than \(l\). To guarantee the completeness, the requirements should be relaxed. Specifically, for each record \(o\) in \(S\) that cannot be identified by any valid PR with length no more than \(l\), a valid PR with the smallest length that identifies \(o\) should be discovered; if no valid PR can identify \(o\), a valid NR with the smallest length that identifies \(o\) should be discovered.

The following propositions (see Appendix for proofs) prove the hardness of our problem. Without loss of generality, it is assumed that there exists no primary key in the data set.

Proposition 4. Given a length-threshold \(l\) and a data set \(S\), for any record \(o\) in \(S\), determining whether there exists a valid PR
Proposition 5. Given a data set $S$, for any record $o$ in $S$, finding a valid PR $r$ with the smallest length that identifies $o$ (we call this problem “SPR” for brief) is NP-Hard.

This proposition can be easily proved from Proposition 4.

Proposition 6. Given a data set $S$, for any record $o$ in $S$, finding a valid NR $r$ with the smallest length that identifies $o$ (we call this problem “SNR” for brief) is NP-Hard.

### 3.1.2 Framework

Given a record $o$ in $S$, since finding a valid PR (NR) with the smallest length that identifies $o$ is NP-hard, to make the problem tractable, we relax the requirements to find a minimal valid PR (NR) that identifies $o$.

**Definition 11 (sub-rule).** ER-rule $r_1$ is a sub-rule of ER-rule $r_2$ if $\text{RHS}(r_1) = \text{RHS}(r_2)$ and the set of the clauses in LHS($r_1$) is a proper subset of the clauses in LHS($r_2$).

**Definition 12 (minimal rule).** ER-rule $r$ is a minimal rule for data set $S$, if Cov($r$) is a sub-rule of Cov($r'$) for any sub-rule $r'$ of $r$, where $S(o) = \{o \in S \land o \text{ identifies } \text{RHS}(r)\}$.

Example 6. Consider the following rules. As can be seen, $r_2$ is a sub-rule of $r_1$. $r_1$ is not a minimal rule because $\text{Cov}(r_1) \subseteq S_1$. In contrast, $r_2$ is a minimal rule.

Clearly, $r$ is a minimal valid rule for $S$, if $r$ is valid for $S$ and $r'$ is not valid for $S$ for any sub-rule $r'$ of $r$.

Now we present the framework of ER-rule set discovery (shown in Algorithm 1), denoted by Dscr.

**Algorithm 1 Framework of Rule Discovery (Dscr)**

**Input:** length-threshold $l$, training data set $S = \{S_1, \ldots, S_m\}$

**Output:** ER-rule set $R$

1. $S \leftarrow S_1 \cup \ldots \cup S_m$;
2. $R \leftarrow \text{Gen-PR}(l, S)$;
3. if Cov($r$) $\neq S$ then
4. $S' \leftarrow S \setminus \text{Cov}(r)$;
5. for each $o$ in $S'$ do
6. if $r_o$ is valid then
7. $R_i$.insert(MIN-RULE($r_o$));
8. else
9. $R_i$.insert(GEN-SINGLENR($o$));
10. end if
11. end for
12. end if
13. $R_{\text{min}} \leftarrow \text{GREEDY-SETCOVER}(R, S)$;
14. Return $R$;

Given a training data set $S$ and a length-threshold $l$, the valid PRs with length no more than $l$ are generated at first by Gen-PR (line 2), denoted by $R$.

If $R$ is not complete for $S$ (line 3), for each record $o$ that is not covered by $R$, if the corresponding rule $r_o : \land_{i \in k} \land_{j \in l} \Rightarrow e_o$ is valid, a minimal valid sub-rule of $r_o$ is generated by Min-Rule and is added into $R$ (line 7); otherwise, according to Proposition 7 there exists no valid PR that identifies $o$, then a minimal valid NR to identify $o$ is generated by Gen-SINGLENR and added into $R$ (line 9).

Finally, to ensure the independence (see Proposition 8) and reduce the number of rules to accelerate ER, after a valid and complete ER-rule set $R$ is generated, a minimal subset $R_{\text{min}}$ of $R$ is generated by applying the greedy algorithm GREEDY-SETCOVER [34] for the set covering problem, where $S$ is the universe to be covered and $R$ is the collection of subsets of $S$ in which each rule $r$ in $R$ is mapped to the subset $\text{Cov}(r)$ (line 10). During the greedy selection step, for two rules $r_1$ and $r_2$ with the same incremental gain, we choose the rules based on the following priority: if $r_1$ is PR and $r_2$ is NR, $r_1$ is selected; or if $r_1$ and $r_2$ are both PRs (NRs), but $|r_1| \leq |r_2|$, $r_1$ is selected.

**Proposition 7.** Given a data set $S$, for any record $o$ in $S$, there exists a valid PR $r$ that identifies $o$ iff $r_o : \land_{i \in k} \land_{j \in l} \Rightarrow e_o$ is valid.

**Proposition 8.** If $R$ is minimal for data set $S$, then $R$ is independent.

To support the algorithm, we use $(r, d, e, d, \text{exp}, \text{coverage})$ to describe each rule $r$, where $r$ is the id of $r$, $d$ is $\text{RHS}(r)$, $e$ is $\text{LHS}(r)$, and coverage is the set of records covered by $r$. A hash table $H$ is maintained to store the information of coverage. Specifically, given $(r_i, e_j)$ as the key, $H(r_i, e_j) = \text{Cov}(r_i) \cap S_j$. For instance, in Table 2, $H(r_1, e_2)$ returns the records in $S_2$ which match LHS($r_1$). Each time a new rule is generated, the hash table is used for checking its validity.

### 3.2 Gen-PR

We first define the following concepts and propositions for the convenience of discussion.

**Definition 13 (preliminary rule).** $r$ is called a preliminary rule for $S$, if $r$ is not valid for $S$ but $r$ identifies at least one record in $S$.

According to Proposition 3, if $\text{RHS}(r) = e_j$, $r$ is a preliminary rule for $S$ iff $\text{Cov}(r) \notin S_i$ and $\text{Cov}(r) \cap S_j = \emptyset$.

Since a valid ER-rule with empty coverage does not need to be discovered, we require that the coverage of each valid ER-rule is not empty.

**Proposition 9.** If $r$ is a minimal valid rule for $S$ and $|r| > 1$, then any sub-rule of $r$ must be a preliminary rule of $S$.

**Definition 14 (conjunction).** Let $r_1$, $r_2$ be two ER-rules in which $\text{RHS}(r_1) = \text{RHS}(r_2)$. The conjunction of $r_1$ and $r_2$ is an ER-rule $\text{LHS}(r_1) \land \text{LHS}(r_2) \Rightarrow \text{RHS}(r_1)$, denoted as $r_1 \land r_2$.

For simplicity, $r$ is called an “atomic” rule if its length equals to 1. $r$ is called a rule of entity $e_j$, if $\text{RHS}(r) = e_j$ and $r$ is called a rule set of entity $e_j$, if each rule in $R$ is a rule of $e_j$.

Since minimal valid rules can be generated by conjunctions of preliminary rules according to Proposition 9, Gen-PR first generates all the atomic PRs and then iteratively generates PRs with length of $k + 1$ by conjunctions of preliminary PRs of length $k$ and preliminary atomic PRs.
In our algorithm (shown in Algorithm 2), for each entity \(e_j\), the generated preliminary PRs of \(e_j\) are stored in \(L_j^P\), the preliminary atomic PRs of \(e_j\) are stored in \(L_j^A\), and valid PRs of \(e_j\) are stored in \(R_j\).

**Algorithm 2 Gen-PR**

**Input:** \(l, S\)

1. for each \(S_i\) in \(S\) do
2. for each attribute-value pair \(t\) in \(S_j\) do
3. if \(Cov(t) \subseteq S_j\) then
   4. \(R_j\).insert(\(r(t)\));
5. else
6. \(L_j^P\).insert(\(r(t)\));
7. \(L_j^A\).insert(\(r(t)\));
8. end if
9. end for
10. for each \(r_k\) in \(L_j^P\) do
11. if \(|r_k| > l\) then
12. break;
13. end if
14. for each \(r_k\) in \(L_j^P\) do
15. if \(Cov(r_k \wedge r_k) \neq \emptyset\) then
16. if \(Cov(r_k \wedge r_k) \subseteq S_j\) then
17. \(R_j\).insert(\(r_k \wedge r_k\));
18. else
19. \(L_j^P\).insert(\(r_k \wedge r_k\));
20. end if
21. end if
22. end for
23. end for
24. end for
25. \(R = R_1 \cup R_2 \cup \cdots \cup R_{|S|}\)
26. Return \(R\);

**Step 1. Generate atomic PRs (lines 2-9).** To find all the preliminary and valid atomic PRs of \(e_j\), for each attribute-value pair \(t\) that appears in the records in \(S_j\), we should check whether the corresponding rule \(r(t): t \Rightarrow e_j\) is preliminary or valid. Specifically, if \(r(t) \subseteq S_j\), then \(r(t)\) is valid (according to Proposition 3) and is added to \(R_j\) (lines 3-4); otherwise \(r(t)\) is a preliminary and \(r(t)\) is added to both \(L_j^P\) and \(L_j^A\) for further conjunction (lines 5-7).

**Step 2. Generate PRs with length \(>1\) (lines 10-23).** In this step, we conjunct each preliminary rule \(r_i\) in \(L_j^P\) (line 8) with each preliminary rule \(r_k\) in \(L_j^P\) (line 9) to generate a new PR, \(r_i \wedge r_k\). This new rule is added to \(R_j\) if it is valid and its coverage is not empty (lines 10-12); or added to \(L_j^P\) for further conjunctions if it is preliminary (lines 13-14); otherwise it is useless and can be ignored.

In contrast to traditional rule generation method, such as FOIL [42], where each rule is generated by iteratively adding the best literal into the rule until the rule becomes valid, we generate each rule by enumerating all possibilities. The reason why we do not use the greedy strategy similar as FOIL is that our problem does not satisfy the greedy selection property. That is, the best literal does not have to be included in the optimal result.

Now we use an example to illustrate the process of Gen-PR.

**Example 7.** Consider \(o_{11} \supseteq o_{12}\) in Table 1. Let the operator for \(ca\) be \(\epsilon\). Suppose “\(\epsilon\)” is also in \(o_1[\epsilon]\).

**First, we generate the valid PR set \(R_1\) to identify records referring to \(e_1\). The steps of generating \(R_1\) are shown in Table 3. In Step 1, atomic PRs (\(r_1, r_2, r_3\) and \(r_4\)) are generated. Among these rules, \(r_3, r_4\) are stored in \(R_1\) since they are valid, and \(r_1, r_2\) are stored in both \(L^P_1\) and \(L^A_1\) since they are preliminary. In Step 2, we conjunct rules in \(L^P_1\) with rules in \(L^A_1\) (\(L^P_1 = L^A_1 = \{r_1, r_2\}\)). Then a new rule, \(r_1 \wedge r_2\), is generated. \(r_1 \wedge r_2\) is verified to be valid and is inserted into \(R_1\). In this way, the PR set \(R_1\) of \(e_1\) is generated. Similarly, \(R_2\) and \(R_3\) are generated. Finally, the algorithm outputs the result, \(R_1 \cup R_2 \cup R_3\).

The following Lemma shows the correctness of Algorithm 2.

**Lemma 1.** Given a length-threshold \(l\) and a training data set \(S\), the result \(R\) output by Gen-PR is a valid PR set in which the length of each rule is no more than \(l\).

**3.3 Gen-SingleNR**

**Gen-SingleNR** composes two steps. Given a record \(o\), a valid NR \(r\) that identifies \(o\) is generated at first by **Sel-Clauses**; then a minimal sub-rule of \(r\) is output by **Min-Rule**.

**Sel-Clauses.** To find a valid rule \(r\) that identifies \(o\), **Sel-Clauses** works as follows. First, \(r\) is initialized as \(r_o\) (\(r_o: \land_{i \in e_i} t_i \Rightarrow o\)); then for each record \(o_i\) in \(Cov(r_o)\backslash S_o\) (\(S_o\) denotes the cluster in \(S\) that includes \(o\)), a negative clause \(T_i\) which is satisfied by \(o\) and not satisfied by \(o_i\) is found and added into \(LHS(r)\). This step is shown in Algorithm 3.

**Algorithm 3 Sel-Clauses**

**Input:** record \(o\)

1. \(T \leftarrow o\);
2. for each \(o_i\) in \(Cov(r_o)\backslash S_o\) do
3. for each \(t_j\) in \(o_i\) do
4. if \(o\) satisfies \(\neg t_j\) then
5. \(T\).insert(\(\neg t_j\));
6. end if
7. end for
8. end for
9. Return \(\land_{T_j \in T} T_j \Rightarrow o\);

Proposition 10 shows the correctness of Algorithm 3 (proof is in Appendix).
TABLE 4
An Example of Gen-singleNR

<table>
<thead>
<tr>
<th>record</th>
<th>entity</th>
</tr>
</thead>
<tbody>
<tr>
<td>o₁₁ = {t₁, t₂}</td>
<td>e₁</td>
</tr>
<tr>
<td>o₁₂ = {t₁, t₂, t₃}</td>
<td>e₂</td>
</tr>
<tr>
<td>o₂₂ = {t₁, t₂, t₄}</td>
<td>e₂</td>
</tr>
</tbody>
</table>

Proposition 10. Given a record o, Sel-CLAUSES outputs a valid ER-rule r that identifies o.

MIN-RULE. Given an ER-rule r, MIN-RULE (shown in Algorithm 4) checks each clause Tᵢ in r (line 1) to identify whether Tᵢ can be removed (line 3). If so, Tᵢ is removed from r (line 4).

Algorithm 4 Min-Rule
Input: ER-rules r : T₁ ∧ T₂ ... ∧ Tₖ ⇒ e_j
1: for each clause Tᵢ in r do
2: let r' be the sub-rule of r that excludes Tᵢ;
3: if Cov(r') \ Sⱼ = Cov(r') \ \ Sⱼ then
4: r ← r';
5: end if
6: end for
7: Return r;

Proposition 11 shows the correctness of Algorithm 4 (proof is in Appendix).

Proposition 11. Given a valid ER-rule r, Min-Rule outputs a minimal sub-rule r’ of r.

The following lemma can be easily proved from Proposition 10 and Proposition 11.

Lemma 2. Given a record o, the result r output by Gen-singleNR is a minimal valid NR that identifies o.

We use an example to illustrate the whole process of Gen-singleNR.

Example 8. As shown in Table 4, there are four records, o₁₁, o₁₂, o₂₂ and o₁₁. Suppose the operator for each attribute is =. We want to find a minimal valid NR that identifies o₁₁.

Sel-CLAUSES. First we initialize r as t₁ ∧ t₂ ⇒ e₁. Since Cov(r) = {o₁₁, o₂₁, o₂₂} and S₁ = {o₁₁}, Cov(r) \ S₁ = {o₂₁, o₂₂}. For all the attribute-value pairs in o₂₁, only t₃ does not appear in o₁₁, then o₁₁ satisfies ¬t₃. Thus we add ¬t₃ into LHS(r). For o₂₂, as t₄ does not appear in o₁₁, we add ¬t₄ into LHS(r). Now a valid rule r that identifies o₁₁ is generated such that r = t₁ ∧ t₂ ∧ ¬t₃ ∧ ¬t₄ ⇒ e₁.

MIN-RULE. For each clause Tᵢ in LHS(r), we check whether the sub-rule of r excluding Tᵢ is still valid. Since t₂ ∧ ¬t₃ ∧ ¬t₄ ⇒ e₁ is still valid, t₁ is removed. Since ¬t₃ ∧ ¬t₄ ⇒ e₁, t₂ ∧ ¬t₄ ⇒ e₁ and t₂ ∧ ¬t₄ ⇒ e₁ are all invalid, t₂, t₃ and t₄ should not be removed. Thus, the minimal sub-rule t₂ ∧ ¬t₃ ∧ ¬t₄ ⇒ e₁ is output.

3.4 Analysis

The following theorem shows the correctness of Algorithm 1.

Theorem 2. Given a training data set S, Algorithm 1 outputs an ER-rule set Rₘᵢₙ that is valid, complete and independent.

Now we analyze the time complexity of our algorithms.

In this section, we discuss the algorithm of entity resolution by leveraging ER-rules. We first define the rule-based ER problem. Next we develop an online algorithm for rule-based ER problem. Finally, we describe how to incorporate this algorithm into a generalized ER framework.

Problem 1 (Rule-based ER). Rule-based ER takes U and Rₑ as input, and outputs U. U is a data set, Rₑ is an ER-set of entity set E = {e₁, ..., eₘ}, U = {U₁, ..., Uₘ} is a partition of records where each group Uⱼ (1 ≤ j ≤ m) is a subset of U which are determined to refer to the entity eⱼ and U₁≤j≤m Uⱼ is a subset of U.

Our rule-based ER algorithm R-ER (shown in Algorithm 5) scans records one by one and determines the entity for each record. The determination process can be divided into the following steps.

First, we find all the rules satisfied by o (FindRules). Second, for each entity e to which o might refer, we compute the confidence that o refers to e according to the rules of e that are satisfied by o (ConfConf). Third, we select the entity e with the largest confidence to which o might refer, and if this confidence is larger than a confidence threshold, it is determined that o refers to e (SelEntity). These procedures are described as follows.

FindRules (lines 14-25). It takes a record o and find all the rules that are satisfied by o. The intuitive idea is to compare o with each rule. In practice, o does not match the LHS of most of the rules. In order to find the rules whose LHS are matched by o, we construct an inverted index and a B-tree for rules, denoted by Lₑ and Tₑ respectively. Lₑ is for rules including clauses with = and ∈ operators and Tₑ is for rules including clauses with range operators, such as ≤. For example, given an attribute-value pair t = (name,”wang”), Lₑ(t) or Tₑ(t) stores the rules which include t.
Algorithm 5 R-ER

Input: $U$, $R_E$, $\theta_C$
Output: $U$
1: Initialize:
2: for each entity $e_j$ in $E$ do
3: \hspace{1em} $U_j \leftarrow \emptyset$;
4: end for
5: for each $o_i$ in $U$ do
6: \hspace{1em} $R(o_i) \leftarrow$ FINDRULES($o_i$);
7: for each attribute-value pair $r$ in $o_i$ do
8: \hspace{2em} $R(o_i) \leftarrow R(o_i) \cap L_R(t) \cap T_R(t)$;
9: end for
10: for each $r$ in $R(o_i)$ do
11: \hspace{2em} if $o_i$ does not match LHS($r$) then
12: \hspace{3em} $R(o_i) \leftarrow R(o_i) - \{r\}$;
13: \hspace{2em} end if
14: end for
15: end procedure
16: procedure FINDRULES($o_i$)
17: \hspace{1em} $R(o_i) \leftarrow \emptyset$;
18: for each attribute-value pair $t$ in $o_i$ do
19: \hspace{2em} $R(o_i) \leftarrow R(o_i) \cap L_R(t) \cap T_R(t)$;
20: end for
21: for each $r$ in $R(o_i)$ do
22: \hspace{2em} if $o_i$ does not match LHS($r$) then
23: \hspace{3em} $R(o_i) \leftarrow R(o_i) - \{r\}$;
24: \hspace{2em} end if
25: end for
26: end procedure
27: procedure COMPCONF($R$)
28: \hspace{1em} $C \leftarrow \sum_{r \in R} w(r)$;
29: return $C$;
30: end procedure
31: procedure SELENTITY($o_i$, $\theta_C$)
32: \hspace{1em} $j' \leftarrow \arg\max \{C(o_i, e_j) | 1 \leq j \leq m\}$
33: if $C(o_i, e_j') \geq \theta_C$ then
34: \hspace{2em} add $o_i$ to $U_{j'}$;
35: end if
36: end procedure

Example 9. Let us consider record $o_1$ in Table 1 and the rule set in Example 2. An inverted index for rules is shown in Table 5. As $o_1$ is named “wei wang” and has a coauthor “zhang”, we first find the related rules of ($name, "wei wang""); then we find the related rules of ($coa, "zhang"$) which is $\{r_1\}$; since the intersection of the two rule sets is $\{r_1\}$, $o_1$ should compare with only one rule, $r_1$.

COMPCONF (lines 26-29). Let $R(o)$ denote the rule set satisfied by $o$ and $R(e)$ denote the rule set of entity $e$. Intuitively, the more rules in $R(o) \cap R(e)$, the larger the weight $w(r)$ of each rule $r$ in $R(o) \cap R(e)$, the more confident that $o$ refers to $e$. Thus the confidence of $o$ referring to $e$ (denoted by $C(o, e)$) is defined as below. \[ C(o, e) = \sum_{r \in R(o) \cap R(e)} w(r). \]

Example 10. Consider $o_{2_{w}}$ in Table 1, the rule set in Example 2 and another rule $r_5$: ($name, "wei wang"$)$ \wedge (coa, "duncan") \Rightarrow e_1$. Let the weight of each rule be 1. By running FINDRULES on $o_{2_{w}}$, we can get $R(o_{2_{w}}) = \{r_1, r_5\}$. Since $R(e) = \{r_1, r_4, r_5\}$, the confidence results for $o_{2_{w}}$ are $C(o_{2_{w}}, e_1) = w(r_1) + w(r_5) = 2$; $C(o_{2_{w}}, e_2) = C(o_{2_{w}}, e_3) = 0$.

SELENTITY (lines 30-35). Given a record $o_i$, we select the entity $e_j$ which has the largest confidence among all the entities. If the largest confidence $C(o_i, e_j)$ is also larger than a given confidence threshold, then $o_i$ is determined to refer to $e_j$ and is put into the group $U_j$; otherwise, it is determined that $o_i$ does not refer to any entity in $E$.

Example 11. Let us consider the confidence results for $o_{12}$ in Example 11. Let $\theta_C = 0$. As $C(o_{12}, e_1) = 2$ is the maximum and $C(o_{12}, e_1) > \theta_C$, $o_{12}$ is then determined to refer to $e_1$ and is put into group $U_1$.

We now analyze the complexity of Algorithm 5.

Theorem 4. Let $n_o$ be the number of records, $n_r$ be the maximal size of related rule set for each record, $l_m$ be the maximum length of rules, and $l_a$ be the average number of attribute-value pairs for each record. Algorithm 5 runs in $O(n_o \cdot n_r \cdot l_m \cdot l_a)$ time.

In practice, $l_m$, $l_a$ and $n_r$ are quite small which can be considered as constants, then the running time of Algorithm 5 is linear to the number of records.

Application in generic ER. Now we present a generic ER framework, shown in Algorithm 6, where our approach is combined with traditional ER methods to resolve entities.

Algorithm 6 GENERAL-ER

Input: $U$, $R$, $E$
Output: A partition $U$ of $U$
1: $U' = \text{R-ER}(U, R)$;
2: $U'' = \text{MERGE}(U')$;
3: $U = \text{T-ER}(U \cup U'' - U_1 \cup V \cup U_1)$;
4: Return $U$

Given a data set, we first use existing ER-rules to identify records (line 1). These rules can be discovered from existing high quality data such as master data or manually identified data. Inspired by the swoosh method [21], each cluster is then merged into a composite record via a merge function (line 2). Finally a traditional ER method, denoted by T-ER, can be applied (line 3) to identify the new data set.

Moreover, in order to identify more records, the current ER result can be used as the training data to discover new ER-rules. The training data can also be obtained by using techniques, such as relevant feedback, crowd sourcing and knowledge extraction from the web. Therefore, with the accumulated information, ER-rules for more entities can be discovered.

5 Rule Update

The discovered rule set might be invalid, incomplete, or contain useless rules if the training data is incomplete or out-of-date. To
ensure the performance of the discovered rule set on new records, we introduce an evolution method of rules in this section. First, we introduce the problems that rules might have.

Invalid rule. A rule \( r \) is invalid if there exist records that match \( \text{LHS}(r) \) but do not refer to \( \text{RHS}(r) \). Invalid rules might be discovered when the information of entities is not comprehensive. For example, suppose the training data set involves the records in Table 1 except \( o_{31} \). The rule \( r \) (name = “wei wang”) \& (coa \( \in \) “zhang”) \( \Rightarrow e_1 \) can be generated. For \( o_{31} \), it matches \( \text{LHS}(r) \) but does not refer to \( e_1 \). Therefore, \( r \) is an invalid rule.

Useless rule. An ER-rule \( r \) is called a useless rule if \( \text{ Cov}(r) = \emptyset \), since no records are identified by \( r \). Rules will become useless when entity features change. For instance, authors may change their research areas or teams. As a result, the topics, conferences and coauthors of their papers will change correspondingly. Then some rules may become useless. For example, the author \( \text{wei wang} \) in UNSW has changed his research interest from \( \text{XML} \) to \( \text{keyword search} \) and \( \text{similarity join} \); and his coauthors from \( \text{ju} \) and \( \text{jian} \) to \( \text{lin} \), etc.

Incomplete rule set. An ER-rule set \( R \) of entity set \( E \) is incomplete if there are records referring to entities in \( E \) that are not covered by \( R \). Both the incomplete information of entities and continuous changes of entity features would cause a rule set become incomplete.

To solve these problems, we develop some methods to identify candidate invalid rules and candidate useless rules and discover new effective ER-rules.

Identify invalid rules. Rules \( r_1 \) and \( r_2 \) are candidate invalid rules if there is a record \( o \) that matches both \( \text{LHS}(r_1) \) and \( \text{LHS}(r_2) \) but \( \text{RHS}(r_1) \neq \text{RHS}(r_2) \).

Identify useless rules. Given a time threshold \( \theta (\text{days}) \) and a number threshold \( \theta_n \), we determine a rule \( r \) to be a candidate useless rule if for \( \theta (\text{days}) \), more than \( \theta_n \) new records have been determined to refer to \( \text{RHS}(r) \) but none of them is identified by \( r \).

Discover new rules. By considering the R-ER result of a new data set as a training data, the rule discovery strategy can be applied to discover new rules.

After candidate invalid rules, candidate useless rules or new rules are discovered, by exploiting users’ feedback, we can finally determine among these rules, which should be deleted or inserted and the rule set can then be updated accordingly.

6 EXPERIMENTAL EVALUATION

In this section we perform extensive experiments to validate our methods. Using real data sets, we evaluate (1) the effectiveness of our rule learning algorithm (DiscR) and our rule-based ER approach, (2) the impact of training data size on ER accuracy and the number of generated rules, (3) the impact of rule length threshold on ER accuracy, and (4) the scalability of DiscR and R-ER with the size of data.

Note that, to evaluate the effectiveness of both DiscR and R-ER, we compute the ER-results output by R-ER, using the rules discovered by DiscR.

6.1 Experimental Setting

Considering paper-author identification is one of the most difficult ER problems, we use the following data sets to evaluate our algorithms.

(1) dblp data is a selection from DBLP Bibliography\(^3\) containing 1,812 paper-authors, which is divided into groups according to the authors’ identities in DBLP. The author names in this data set (shown in Table 6) are quite representative, since each name is shared by a large number of authors. Hence, these records might be the most difficult to identify in DBLP.

(2) kdd data\(^4\) is the validation data set for Track 1 of KDD-Cup 2013, which contains 47,081 paper-authors. It is the ground truth obtained from the user edits at the Academic Search website, where an assignment of a paper to an author is known to be incorrect if an author deleted the paper from the profile, or correct if an author confirmed it [40].

The overlap between these two data sets is little since they only have 26 paper-authors in common, which is 0.05 percent of kdd data and 1.4 percent of dblp data.

(3) training data are produced by random sampling from dblp data and kdd data, controlled by the parameter train\%: the percentage of records sampled from each cluster. For instance, the training data with train\% = 20\% is generated by sampling max\{0.2|e|, 1\} records from each cluster \( e \) in the original data. Note that, there is at least one selected record for each cluster.

To test the effectiveness of our rule discovery algorithm, the average cluster size in the training data should be small, such as \( \leq 3 \). Hence, we fix train\% = 20\% for dblp data and train\% = 5\% for kdd data due to their different average cluster sizes.

Algorithms. We compare our proposed algorithms against GHOST [11]. GHOST is one of the state-of-the-art name disambiguation algorithms which focuses on the problem of identifying authors with identical names in publications.

The basic idea of GHOST is to build a graphical model of the input such that each node represents a record and each edge represents a co-authorship and then computes similarities between records by exploiting the relationships among every pair of publications.

We also compared our method against the leading algorithm [41] in the KDD cup 2013, denoted by CFR. This algorithm extracted several features from the provided data set at first, and then trained classification and ranking models using these features, and finally combined these models to boost the performance.

Since most of the paper-authors can be identified based on the author name and one of the coauthors, we set the rule length threshold \( l = 2 \) by default. To permit the match between a full name and its abbreviation, we define the operator for attribute name as a fuzzy matcher \( \approx \), such that two names are matched only if one name can be

<table>
<thead>
<tr>
<th>name</th>
<th>#aut</th>
<th>name</th>
<th>#aut</th>
</tr>
</thead>
<tbody>
<tr>
<td>“jing wang”</td>
<td>11</td>
<td>“ping zhang”</td>
<td>4</td>
</tr>
<tr>
<td>“yan liu”</td>
<td>10</td>
<td>“hui wang”</td>
<td>11</td>
</tr>
<tr>
<td>“jian zhang”</td>
<td>8</td>
<td>“xin zhang”</td>
<td>13</td>
</tr>
<tr>
<td>“lei chen”</td>
<td>4</td>
<td>“jun yang”</td>
<td>9</td>
</tr>
<tr>
<td>“jun sun”</td>
<td>7</td>
<td>“wei wang”</td>
<td>25</td>
</tr>
</tbody>
</table>

transformed into the other by inserting several letters (neither deletion nor replacement is allowed). Specifically, \(s_1 \approx s_2\) if edit-distance\((s_1, s_2) < 1\), where the costs of insertion, deletion and replacement are 0.1, 1 and 1 respectively.

Our algorithms were implemented in C++ and compiled using Microsoft Visual Studio 2010. The experiments are conducted on a core i7 2.00 GHz PC with 8GB RAM, running Microsoft Windows 7.

**Accuracy Measures.** To ensure a fair comparison with other ER approaches, two accuracy measures are used. One is F-measure used by GHOST, which is the harmonic mean of precision and recall. The other is mean average precision (MAP) used by CFR, which is also a well-known measure from information retrieval that factors in precision at all recall levels [40].

### 6.2 Comparison

In the first set of experiments, we compare the effectiveness of our methods with GHOST and CFR. The comparison results are reported in Table 7. We have the following observations. (1) R-ER outperforms GHOST and CFR by up to 41 and 1 percent, respectively. This verifies the benefits of both our rule-based entity resolution method and our rule discovery algorithm on the effectiveness. (2) The lowest F-measure of R-ER is 76 percent, while it is 28 percent for GHOST. This result verifies the robustness of our method.

### 6.3 Effect of Updating Rules

In this section, we evaluate the impact of updating rules on the accuracy of ER-result. We compare the accuracy of ER-

### Table 7

<table>
<thead>
<tr>
<th>F-measure</th>
<th>GHOST</th>
<th>R-ER</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;hui wang&quot;</td>
<td>0.32</td>
<td>0.92</td>
</tr>
<tr>
<td>&quot;jian zhang&quot;</td>
<td>0.28</td>
<td>0.76</td>
</tr>
<tr>
<td>&quot;jing wang&quot;</td>
<td>0.95</td>
<td>0.98</td>
</tr>
<tr>
<td>&quot;jun sun&quot;</td>
<td>0.95</td>
<td>1.00</td>
</tr>
<tr>
<td>&quot;jun yang&quot;</td>
<td>0.75</td>
<td>0.97</td>
</tr>
<tr>
<td>&quot;lei chen&quot;</td>
<td>0.23</td>
<td>0.93</td>
</tr>
<tr>
<td>&quot;yan liu&quot;</td>
<td>0.42</td>
<td>0.86</td>
</tr>
<tr>
<td>&quot;ping zhang&quot;</td>
<td>0.28</td>
<td>0.99</td>
</tr>
<tr>
<td>&quot;xin zhang&quot;</td>
<td>0.93</td>
<td>1.00</td>
</tr>
<tr>
<td>&quot;wei wang&quot;</td>
<td>0.43</td>
<td>0.90</td>
</tr>
<tr>
<td>average</td>
<td>0.35</td>
<td>0.93</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MAP</th>
<th>GHOST</th>
<th>R-ER</th>
</tr>
</thead>
<tbody>
<tr>
<td>dblp</td>
<td>0.983</td>
<td>0.989</td>
</tr>
</tbody>
</table>

### Table 8

<table>
<thead>
<tr>
<th>F-measure</th>
<th>no-upt</th>
<th>upt</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;hui wang&quot;</td>
<td>0.85</td>
<td>0.92</td>
</tr>
<tr>
<td>&quot;jian zhang&quot;</td>
<td>0.70</td>
<td>0.76</td>
</tr>
<tr>
<td>&quot;jing wang&quot;</td>
<td>0.97</td>
<td>0.98</td>
</tr>
<tr>
<td>&quot;jun sun&quot;</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>&quot;jun yang&quot;</td>
<td>0.95</td>
<td>0.97</td>
</tr>
<tr>
<td>&quot;lei chen&quot;</td>
<td>0.81</td>
<td>0.93</td>
</tr>
<tr>
<td>&quot;yan liu&quot;</td>
<td>0.77</td>
<td>0.86</td>
</tr>
<tr>
<td>&quot;ping zhang&quot;</td>
<td>0.96</td>
<td>0.99</td>
</tr>
<tr>
<td>&quot;xin zhang&quot;</td>
<td>0.96</td>
<td>1.00</td>
</tr>
<tr>
<td>&quot;wei wang&quot;</td>
<td>0.88</td>
<td>0.90</td>
</tr>
<tr>
<td>average</td>
<td>0.88</td>
<td>0.93</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MAP</th>
<th>no-upt</th>
<th>upt</th>
</tr>
</thead>
<tbody>
<tr>
<td>dblp</td>
<td>0.986</td>
<td>0.989</td>
</tr>
</tbody>
</table>

results based on the rules which are not updated and the rules which are updated. The results are reported in Table 8, where "no-upt" represents the R-ER results with rules not updated, and "upt" represents the R-ER results with the rules updated automatically based on testing data sets without correctness check by users. It is observed that upt outperforms no-upt on both dblp and kdd by up to 5.4 and 0.3 percent respectively. Specifically, upt outperforms no-upt for all the cases on dblp. This verifies that updating rules is indeed useful for improving the quality of rules.

### 6.4 Effect of Training Data Size and Threshold \(l\)

In the first experiment, we examine the effect of varying the training data size on accuracy. The results are reported in Fig. 1a. We have following observations. (1) The accuracy on dblp reaches 90 percent when train\% = 10\%, and the accuracy on kdd reaches 98.8 percent when train\% = 5\%, which shows that our method can achieve high accuracy under a small training data. (2) With the growth of the training data size, although there is a narrow fluctuation caused by random selection of training data sets, the accuracy increases gradually as expected.

In the second experiment, we examine the effect of varying the training data size on the number of generated rules. The results are reported in Figs. 1c and 1d. It shows that the number of rules is larger than the number of training records on both data sets. However, the size of rules would not be large since it grows slowly with the training data size.

In the third experiment, we compare the accuracy as the length threshold \(l\) varies from 1 to 5. The results are reported in Fig. 1b. It shows that the accuracy reaches to the highest value when \(l = 2\) for both data sets, which means
most rules with length larger than 2 are not effective to identify records in our experiments.

6.5 Efficiency and Scalability
In this section, we investigated the runtime performance of DiscR and R-ER algorithms as we increase the number of records.

Fig. 1e shows the performance of DiscR on kdd. It can be observed that DiscR scales linearly with the number of records. The results tell us although the runtime of DiscR in the worst case is quadratic in number of records, it actually scales quite well in practice.

Fig. 1f shows the performance of R-ER on dblp. As expected, the runtime of R-ER is approximately linear to the number of records which is accordance with our time complexity analysis in Section 4. Thus, the result verifies the scalability of R-ER.

Summary. From the experimental results, we can draw the following conclusions. In our experiments, (a) DiscR and R-ER can achieve a high accuracy using a small training data; (b) updating rules indeed help identify records; (c) the number of generated rules scales well with the training data size on both data sets; (d) rules with length larger than 2 are seldom needed to identify records; and (e) both DiscR and R-ER scales well with the size of data.

7 RELATED WORK
The work on entity resolution can be broadly divided into three categories.

Pairwise ER. Most works on ER focus on record matching [1], which involves comparing record pairs and identifying whether they match. A major part of work on record matching focuses on similarity functions [2], [3], [4], [5]. To capture string variations, [6] proposed a transformation-based framework for record matching. Some machine-learning-based approaches [7], [8] can identify matching strings which are syntactically far apart. Similarity based on record relationships [9], [11] are also proposed to solve the people identification problem.

Since in our work, records are not compared with each other, our work is orthogonal to record matching. However, string similarity functions can be applied to fuzzy match operator (denoted by \( \approx \)) in ER-rules. For example, given a string \( s \), we say \( s \approx \) “wei wang” if the edit distance between \( s \) and “wei wang” is smaller than a given threshold.

Decision trees are employed to teach record matching rules in [10]. However, decision trees cannot be used to discover ER-rules. This is because the domain of the right-hand side of record matching rules is \{yes, no\} (two records are mapped or not mapped), while the domain of the right-hand side of ER-rules is an entity set.

Non-pairwise ER. The research on non-pairwise ER includes clustering strategies [11], [15], [16], [32] and classifiers [18], [19]. Most strategies solve ER based on the relationship graph among records, by modeling the records as nodes and the relationships as edges. Machine learning approaches [12], [31] are also proposed by using global information to solve ER effectively. However, these methods are not suitable for massive data because of efficiency issues. We choose a representative work [11] for comparison.

Scaling. Some other works [13], [20], [22] treat the ER algorithm as black box and focus on developing scalable framework for ER. Indexing techniques used for ER have been surveyed by Christen[23]. [14] focuses on how to update ER results efficiently when ER logic evolves. These techniques are orthogonal to our work and can be applied to accelerate our rule-based ER algorithm.

Note that, among the existing works on pair-wise ER, rule-based approaches [35], [36] are closer to our work. These rules differ from our work as they focus on determining whether two records refer to the same entity while our work focuses on determining whether a record refers to an existing entity.

Our preliminary work [29] proposed rule-based ER and rule discovery strategies. However, the preliminary work only proposed a heuristic method for rule discovery, without efficiency and accuracy guarantee. In this paper, we propose a new definition of rules and effective algorithms for rule discovery.

8 CONCLUSION
This paper developed a class of ER-rules which are capable to describe the complex matching conditions between records and entities. Based on these rules, we developed an ER algorithm R-ER. We experimentally evaluated our algorithms on real data sets. The experimental results show that our algorithm can achieve a good performance both on efficiency and accuracy. For future work, we would like to extend our techniques to more general cases. For instance, how to discover ER-rules when the operator for each attribute is not given? We would also like to consider how to incorporate human resources, such as Crowd, into our rule-discovery framework to improve the quality of rules.

APPENDIX A: PROOFS FOR SECTION 3
A.1 Proof of Proposition 4
Proof. The problem of verifying an ER-rule \( r \) whether it is valid and its length is no more than \( l \) is in NP.

We use a reduction from the Set Cover Problem (SCP) [34]. SCP is known to be NP-Complete. Let \((U, C, k)\) be an instance of SCP, where \(U = \{o_1, o_2, \ldots, o_n\}\) is a finite set of elements and \(C = \{C_1, C_2, \ldots, C_m\}\) is a collection of subsets of \(U\). The question is that does \(C\) contain a cover for \(U\) of size \(k\).

We build an instance of LPR in the following way.

1. Construct a data set \(S = \{o_1, \ldots, o_n, o_{n+1}\}\). Let \(S = \{S_1, S_2\}\) be the training data set of \(S\), where \(S_1 = U - \{o_1, \ldots, o_k\}\) is the data set referring to \(e_1\) and \(S_2 = \{o_{n+1}\}\) is the data set referring to entity \(e_2\).

2. Let \(o_{n+1} = \{t_1, t_2, \ldots, t_m\}\) where each \(t_i\) in \(o_{n+1}\) is an attribute-value pair, such that \(\text{Cov}(t_i) = \{S_1 \setminus C\} \cup \{o_{n+1}\} = (S_1 \setminus C_i) \cup S_2 = S \setminus C_i\). Thus \(\text{Cov}(t_i) = C_i\).

The question is that whether there exists a valid PR \(r\) such that \(r\) identifies \(o_{n+1}\) and \(|r| \leq k\). Clearly, \(r\) is valid and identifies \(o_{n+1}\) iff \(\text{Cov}(r) = \{o_{n+1}\} = S_2\), that is \(\text{Cov}(r) = S \setminus S_2 = S_1\).

If \(C' = \{C_1, C_2, \ldots, C_{i'}\}\) is the solution to the instance of SCP, let \(r\) be the ER-rule \(t_{i_1} \land t_{i_2} \land \ldots \land t_{i_k} \Rightarrow e_2\), then we have,
Thus, \( r \) is the solution to the instance of LPR since \( Cov(r) = \{ o_{i+1} \} \) and \( |r| = k \).

On the contrary, if \( r = t_1 \land t_2 \land \cdots \land t_k \Rightarrow e_2 \) is the solution to the instance of LPR, then \( C' = \{ C_1, C_2, \ldots, C_i \} \) is the solution to the instance of SCP. Clearly, this reduction is polynomial. Hence, LPR is NP-Complete.

A.2 Proof of Proposition 6

**Proof.** It is similar to the proof of Proposition 4. We use a reduction from the smallest set covering problem (SCP). Given an instance \((U, C)\) of SCP, we build an instance of SNR in the following way.

Let \( S = \{ S_1, S_2, S_3 \} \) be the training data set of \( S = \{ o, o', o_1, \ldots, o_n \} \), where \( S_1 = \{ o \} \) is the data set referring to \( e_1 \), \( S_2 = \{ o' \} \) is the data set referring to \( e_2 \) and \( S_3 = U = \{ o_1, \ldots, o_n \} \) is the data set referring to \( e_3 \). Suppose that the operator for each attribute is \( \land \). Let \( T_1, T_2 \) and \( T_3 \) be the sets of attribute-value pairs that appear in \( S_1, S_2 \) and \( S_3 \) respectively, where \( T_1 = \{ t_0 \}, T_2 = \{ t'_0 \} \) and \( T_3 = \{ t_1, t_2, \ldots, t_m \} \). Moreover, for each \( t_i \leq i \leq m \), \( Cov(t_i) = C_i \).

Our goal is to find a valid \( NR \) such that \( r \) identifies \( o \) and \( |r| \) is minimized. Clearly, \( r \) is valid and identifies \( o \) iff \( Cov(r) = \{ o \} \).

If \( C' = \{ C_1, C_2, \ldots, C_i \} \) is the solution to the instance of SCP, let \( r \) be the \( ER \)-rule \( r : t_0 \land \neg t_1 \land \neg t_2 \land \cdots \land \neg t_k \Rightarrow e_1 \), then we have

\[
Cov(r) = Cov(\neg t_k) \cap Cov(\neg t_2) \cdots \cap Cov(t_0) = Cov(t_0) \cup Cov(t_1) \cup \ldots \cup Cov(t_k) \\
= C_1 \cup C_2 \cup \ldots \cup C_i \cup \{ o' \} = U \cup \{ o' \}.
\]

Then we have \( Cov(r) = \{ o \} \). Thus \( r \) is a valid rule with length \( k \) that identifies \( o \). We prove that \( r \) is the solution to the instance of SNR. If the solution to the instance of SNR is not \( r \), but \( r' = t_0 \land \neg t_1 \land \neg t_2 \land \cdots \land \neg t_k \Rightarrow e_1 \), where \( k' < k \). Then \( C' = \{ C_1, C_2, \ldots, C_{k'} \} \) is a smaller subset than \( C' \) that covers \( U \), which contradicts the assumption that \( C' \) is the solution to the instance of SCP. Similarly, if \( r = t_0 \land \neg t_1 \land \neg t_2 \land \cdots \land \neg t_k \Rightarrow e_1 \), the solution to the instance of SNR, by contradiction it can be proved that \( C' = \{ C_1, C_2, \ldots, C_i \} \) is the solution to the instance of SCP. Clearly, this reduction is polynomial. Hence, SNR is NP-Hard.

A.3 Proof of Proposition 8

**Proof.** If \( R \) is minimal for \( S \) and \( R \) is not independent, then there exists a rule \( r \in R \) such that \( R - \{ r \} \not\models r \). Then according to Proposition 1, any record in \( S \) that is identified by \( r \) is also identified by \( R - \{ r \} \), that is \( Cov(r) \subseteq Cov(R - \{ r \}) \). Therefore \( Cov(R) = Cov(R - \{ r \}) \), which contradicts to the assumption that \( R \) is minimal.

Now we prove that \( r \) is valid. \( \forall o_i \in Cov(r) \setminus S_o \), since there exists a clause \( T_k \) in \( \text{LHS}(r) \) such that \( o_i \) does not satisfy \( T_k \), then \( o_i \) does not match \( \text{LHS}(r) \). Thus \( Cov(r) \cap (Cov(r_o) \setminus S_o) = \emptyset \). Then \( Cov(r) \subseteq Cov(r_o) \setminus S_o \).

Since \( r_o \) is a sub-rule of \( r \), \( Cov(r) \subseteq Cov(r_o) \). Then we have the following. Thus, \( r \) is valid.

A.4 Proof of Proposition 11

**Proof.** Clearly, \( r' \) output by Min-Rule is valid. We prove that \( r' \) is minimal. Let \( r = t_1 \land t_2 \land \cdots \land t_m \Rightarrow e \) and \( \text{LHS}(r') = t_1 \land t_2 \land \cdots \land t_{m'} \). If \( r' \) is not minimal, then there exists a clause \( t_i (1 \leq i \leq m') \) in \( r' \) such that the rule \( r'' = t_i \land t_{i+1} \land t_{i+2} \land \cdots \land t_m \Rightarrow e \) is valid. The only reason that \( t_i \) is in \( r' \) is that, the rule \( r'' \); \( t_i \land t_{i+1} \land t_{i+2} \land \cdots \land t_m \Rightarrow e \) is not valid. Since \( r'' \) is a sub-rule of \( r' \), \( r \) should be valid. Hence, the assumption that \( r' \) is not minimal is not true.

A.5 Proof of Theorem 2

**Proof.** From Lemma 1 and Lemma 2, each rule in \( R_{min} \) is valid. Thus \( R_{min} \) is valid. As Algorithm 1 shows (lines 5-11), each of the instances of LPR are solved by M. Therefore, the total running time of LPR is no more than \( \max \{ u \} \).

By Lemma 1 and Lemma 2, each rule in \( R_{min} \) is valid. Thus \( R_{min} \) is valid. As Algorithm 1 shows (lines 5-11), each of the instances of LPR are solved by M. Therefore, the total running time of LPR is no more than \( \max \{ u \} \).

A.7 Proof of Lemma 4

**Proof.** Since \( |Cov(r)\setminus S_o| < n_o \), there are at most \( n_o \cdot l_o \) clauses that should be checked whether they are satisfied by \( o \). Thus Algorithm 3 runs in \( O(n_o \cdot l_o) \).

A.8 Proof of Lemma 5

**Proof.** Since the size of the coverage of each clause in \( r \) is no more than \( n_o \) and each sub-rule of \( r \) has no more than \( |r| \) clauses, the time required for computing the coverage of each sub-rule is no more than \( n_o \cdot |r| \). Thus the running time of Algorithm 4 is \( O(n_o \cdot |r|^2) \).

A.9 Proof of Theorem 3

**Proof.** According to Algorithm 1, the total running time of the whole algorithm is the sum of the following parts (1) the time of GEN-PR (line 2), (2) for all records, the time for checking the validity of \( r_o \) for each \( o \) (line 6) and the maximal time of Min-Rule and GEN-SINGLELR for each \( o \) (line 3-9); and (3) the time of GREEDY-SETCOVER (line 10).

By Lemma 3, the time complexity of part (1) is \( O(n_o \cdot l_o \cdot n_o + n_l) \).

For each record \( o_i \), the time complexity of checking the validity of \( o_i \) is \( O(n_o \times l_o) \). According to Lemma 5, since Min-Rule is
invoked in GEN-SINGLENR, the maximal time complexities of MIN-Rule and GEN-SINGLENR is that of GEN-SINGLENR. That is $O(n_o \cdot |R|^2)$. Thus the time complexity of part (2) is $n_o \times O(n_o \cdot |R|^2)$ since $l_o \leq |R|^2$. With $|R| \leq l_m$, the time complexity of part (2) is $O(n_o^2 \cdot l_m^2)$.

According to Lemma 6, the time complexity of part (3) is $O(n(o, |R| \cdot \min(|R|, n_o))$.

As a sum of the time complexities of these three parts, the total time complexity is $O(n_o, l_o \cdot n_o + n_l^2 \cdot l_m^2 + n_o \cdot |R| \cdot \min(|R|, n_o))$. Thus the time complexity of the algorithm is $O(n_o^2 \cdot l_m + n_l^2 + n_o \cdot |R| \cdot \min(|R|, n_o))$ since $n_c \leq n_o$ and $l_o \leq l_m$.

\section*{ACKNOWLEDGMENTS}
This paper was partially supported by NRF Grant 2012CB316200 and NRF Grant 2012AA011004.

\section*{REFERENCES}
Lingli Li received the master's degree in computer science and engineering from HIT, China. She is currently working toward the PhD degree at Harbin Institute of Technology (HIT). Her research interests include data quality, entity resolution.

Jianzhong Li is a professor and doctoral supervisor at 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 at 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/publications/dlib.