TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 07/13 pp288–302
DOI: 10.26599 / TST.2018.9010004
Volume 23, Number 3, June 2018

Truth Discovery on Inconsistent Relational Data

Jizhou Sun, Jianzhong Li*, Hong Gao, and Hongzhi Wang

Abstract: In this era of big data, data are often collected from multiple sources that have different reliabilities, and there is inevitable conflict with respect to the various information obtained when it relates to the the same object. One important task is to identify the most trustworthy value out of all the conflicting claims, and this is known as truth discovery. Existing truth discovery methods simultaneously identify the most trustworthy information and source reliability degrees and are based on the idea that more reliable sources often provide more trustworthy information, and vice versa. However, there are often semantic constrains defined upon relational database, which can be violated by a single data source. To remove violations, an important task is to repair data to satisfy the constrains, and this is known as data cleaning. The two problems above may coexist, but considering them together can provide some benefits, and to the authors knowledge, this has not yet been the focus of any research. In this paper, therefore, a schema-decomposing based method is proposed to simultaneously discover the truth and to clean the data, with the aim of improving accuracy. Experimental results using real world data sets of notebooks and mobile phones, as well as simulated data sets, demonstrate the effectiveness and efficiency of our proposed method.

Key words: inconsistent data; truth discovery; data cleaning

1 Introduction

Big data provide abundant valuable information but a number of inherent challenges, one of the most serious of these is dealing with the poor quality of data that are often dirty and inconsistent[1]. Even for domains such as flight and stock, which people consider to be highly reliable, a large amount of inconsistency has been observed in data collected from multiple sources[2]. One example is the incorrect news that United Airlines was filing for a second bankruptcy, which sent shares tumbling before the error could be corrected[3]. Decisions based on inconsistent data often result in serious consequences: incorrect flight information may cause passengers to miss flights; errors in stock data may cause financial losses for shareholders; and wrong diagnoses based on incorrect medical measurements are life threatening.

Inconsistencies are typically identified as being violations of data dependencies, such as functional dependencies and conditional functional dependencies. In this respect, two main research areas are committed to solving inconsistency problems, truth discovery, and data cleaning, and the following sections provide a detailed presentation of these areas.

1.1 Truth discovery

It is important that big data are obtained from multiple sources, but due to low reliability of the sources, it is inevitable that inconsistent values will occur for the same object. This kind of inconsistency violates the primary key constraint (a special case of Functional Dependencies, FDs).

For many years, the truth discovery community has

• Jizhou Sun, Jianzhong Li, Hong Gao, and Hongzhi Wang are with School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China. E-mail: sjzh@hit.edu.cn; lijzh@hit.edu.cn; honggao@hit.edu.cn; wangzh@hit.edu.cn.

∗ To whom correspondence should be addressed.

Manuscript received: 2017-07-14; accepted: 2017-08-07

been researching how to obtain the most trustworthy information[4–14]. One common framework used is to estimate the reliability degrees of the sources and determine the most trustworthy claims alternately and iteratively. This means that to determine truths based on the reliability degrees estimated, greater confidence is given to higher degree sources, and based on the truths estimated, sources providing a larger number of correct values are believed to be more reliable. The process is repeated iteratively, until a stopping criterion is met (for example, the maximum number of iterations or no further changes). The intuition behind this framework is that sources that are more reliable often provide more information that is more trustworthy, and that when a source provides consistently trustworthy information it is considered to be more reliable. Within this framework, various studies have adopted different calculation details or have focused on more specified conditions, such as the relationship between sources, relationship between objects, multiple true values, heterogeneous data, and long-tail phenomena.

1.2 Data cleaning

To automatically improve data quality, it is necessary to define data quality rules that determine how to detect and repair errors, and the use of data dependencies (integrity constraints) is the natural choice in this respect[1]. Classical FDs[15] were the first class of dependencies to be introduced, and these are almost as old as the relational database model. However, for data errors to be captured more proficiently, Conditional Functional Dependencies (CFDs) were proposed[16] as an extension of classical FDs, and these incorporate patterns of semantically related data values. Dependencies may be violated when data contain errors, and detecting the errors is tractable[17, 18]. A common heuristic way of removing errors once they have been detected is to instigate a repair involving the use of a modified database, which satisfies the dependencies and uses the least number of updates with respect to the original dirty one. This type of problem is known as the "data cleaning problem". It is intractable, and several methods have been proposed aimed at determining a near optimum repair[19–22] or a set of repair samples[23, 24].

In addition to FDs and CFDs, several other extensions have been proposed, such as extended conditional functional dependencies[25] and conditional dependencies with built-in predicates[26], and new types of rules have been proposed, such as editing rules[27], fixing rules[28], differential dependencies[29], and comparable dependencies[30]. However, as FDs and CFDs are the most common data quality rules, and they have attracted the most research attention, only these two types of rules are considered in this paper; and the others will be considered in future work.

1.3 Problems inherent in previous methods

Truth discovery and data cleaning research groups have provided considerable and valuable research results. However there are several drawbacks remaining that require exploration, and those that are related to this study are presented as follows:

- In most truth discovery methods, all sources start with uniform weights prior to the iteration process. A considerable decrease in performance can result when most sources are unreliable, and therefore, it is extremely important to improve the initialization of source reliability[31].
- In truth discovery, it is common that only a few sources claim for an object, and the discovered values thus have low confidences.
- When dependencies are available, the output of truth discovery methods may still violate the dependencies. A straightforward solution would be to clean the result by these dependencies; however, this may replace some correct answers with incorrect ones.
- The tie phenomenon and cost based data cleaning methods are heuristic and may result in more than one optimum repairs, which means the same number of updates are required in the repairs. This then requires user feedback to determine which one is right, and such an operation involves a nonautomatic cleaning process.
- The errors contained in data do not always violate the dependencies; therefore, the data cleaning methods cannot even detect these errors, let alone fix them.

The authors have observed two problems coexisting in many real-world applications. For example, on an online shopping web site, several sellers may provide conflicting information about the same product, and information from one single seller may violate some dependencies. In another example, systems are developed to extract product specifications on the web (e.g., Ref. [32]), and a product often contains many attributes and dependency relationships between the attributes. In addition, specifications are collected from a large number of web sites that can be considered as data sources, and which have different reliability degrees.

Both truth discovery and data cleaning methods were studied to improve data quality, and when considered together, these methods can complement each other. For example, a source that provides data that are determined to be extremely inconsistent is therefore less reliable and should be initialized with low weight. When the score of a candidate value in truth discovery is associated with the cost in data cleaning, the cost of the data type changes from an integer to a real value, which can eliminate the tie phenomenon. To the author's knowledge, no previous studies have yet taken advantage of using these benefits from both methods.

1.4 Proposed SDTD method

To resolve these problems and exploit the benefits of using both methods, a Schema-Decomposing based Truth Discovery (SDTD) framework is considered, which directly provides results satisfying the dependencies. The initial pre-processing step decomposes claim-tuples into a set of source-key-value triples, which is the input of the truth discovery framework and uses certain decomposing rules related to the dependencies. Each source is initialized with a weight depending on the inconsistent values it provides. The truth discovery iterating process is almost analogous to that used in previous work; however, in this paper dependencies are considered when calculating value scores. After iterations are finished, a post-processing step joins the key-value pairs back into relational tuples, which satisfies the dependencies directly. Another benefit of schema-decomposing is that the claim numbers of some of the objects are greatly increased, and the results determined are more confidential. In summary, this paper makes the the following contributions:

- An SDTD framework is proposed, which seamlessly combines the disciplines of truth discovery and data cleaning. Several benefits can be obtained when using this framework.
- An adequate set of decomposing rules are introduced. According to the rules, all (conditional) FDs are considered, and each claim-tuple can be decomposed into source-key-value triples.
- Dependencies are considered when calculating the scores of candidate values, and more accurate values can be determined.
- In this study, experiments are conducted using two real world data sets and synthetic data sets. The results demonstrate the effectiveness and efficiency of the proposed method.

The remainder of this paper is set out as follows: related research is summarized in Section 2; Section 3 provides a running example to demonstrate our motivation behind this study; Section 4 formulates the problem, the proposed method is derived and analyzed; in Section 5, various experiments are conducted on real world data sets and synthetic data sets to validate the effectiveness and efficiency of the proposed method; and Section 6 presents the conclusions.

2 Related Studies

2.1 Data cleaning

Bohannon et al.[19] modeled the cleaning problem based on cost and provided an effective heuristic algorithm using value modification, and Wijsen[21] proposed a theoretical framework that considers update-based repairing, in which the problem of constructing nucleus (a single database that yields consistent answers to a class of queries without the need for query rewriting) has been studied. More recently, research groups have focused on user participation. For example, Yakout et al. developed a data cleaning system GDR that allows user's guidance[33], and studied the problem of guided data repair[34] where updates are completed in a group-manner.

Xie et al.[35] presented a data cleaning framework that combines data quality rules (CFDS, CINDS, and MDs) with user feedback through an interactive process. Recently, an interactive, deterministic, and declarative data cleaning system, Falcon, was presented[36], which encourages users to explore data, identify possible problems, and develop updates to fix them. Falcon focuses on addressing the challenge of finding a set of SQL update queries that is minimal in size, and it also fixes a large number of errors. Cai et al.[37] proposed a collective inference model of sanitizing data, to prevent social network users from attacks. They also constructed a data-sanitization strategy that can optimize the tradeoff between data utility and privacy in Ref. [38].

Miao et al.[39, 40] studied the insertion propagation problem with functional dependency, plenty of analysis is provided and complexities results are obtained. In Ref. [41], an iterated local least squares imputation method is proposed to estimate the missing values in microarray gene expression data. There are also studies concerning the currency problem of dynamic data, such as Ref. [42].

Other data cleaning research is not covered here due to space limitation, but comprehensive and systematical summaries can be found in the latest survey[43].

2.2 Truth discovery

When the concept of truth discover was first proposed[4], the aim was to determine the truth with respect to conflicting web information. Galland et al.[6] explored techniques based on corroboration and introduced three fix point algorithms corresponding to the different levels of complexity in an underlying probabilistic model. Pasternack and Roth[7] proposed a method, in which prior knowledge was translated into propositional constraints that are integrated into each round of the truth discovery process; however, the use of linear programming to solve the problem is very time consuming. A similar study[44] introduced a new, generalized, fact-finding framework that can incorporate additional information, such as uncertainty, when extracting information that is claimed in documents, such as the attributes of the sources, the degrees of similarity among claims, and the degrees of certainty expressed by sources in the truth discovery process. Furthermore, objects have been clustered together[45], and the truth discovery task is analyzed in a clustering manner. Another study[46] presented, for the first time, a Bayesian interpretation of the basic mechanism used in truth discovery with information networks, and a maximum likelihood estimation approach for truth discovery in social sensing has been proposed[11]. Observations provided by humans can be modeled as binary variables, and the truth discovery task is formulated as a maximum likelihood estimation problem, and solved by an EM algorithm. Truth discovery has been used[13] to resolve conflicts in heterogeneous data (objects with heterogeneous attributes), by truth discovery. And a confidence-aware approach for truth discovery has been proposed[14], in which accuracy was improved by reducing the influence of low confidence sources (where low-confidence means that the sources provide very few claims). Furthermore, situations that have been studied in which more than one value can be true for one object, for example, the authors of a book and the actors in the movie of the book[9, 47].

Other studies (e.g., Refs. [5, 48, 49]) have also focused on the relationship between sources, for example, where one source copies information from another source, and research has also been aimed at selecting with high-quality sources to reduce costs and improve the accuracy of truth discovery methods[50, 51].

Other groups have been committed to determining truths using probabilistic graphical model-based methods. For example, a Bayesian probabilistic approach, GTM, has been proposed that was designed for continuous data type[10]. The same research group also proposed LTM[9] at almost the same time, which is a probabilistic graphical model considering two types of errors under scenarios of multiple truths: false positives and false negatives. A further study used a probabilistic model of Ref. [12], where the source reliability had different semantic meanings in different settings: it could indicate the probability of a source asserting the truth, or the probability of a source both knowing and telling the truth, or even the probability of a source intending to tell the truth.

The latest survey[31] provides more comprehensive introduction to truth discovery.

3 A Motivating Running Example

On a shopping web site, many sellers (data sources) advertise a supply of the same product (an object). Usually, sellers also provide specifications about their goods, which include several to tens of attributes. There are often dependencies between these attributes, and careless sellers may provide wrong information which may violate these dependencies. This situation relates to traditional truth discovery. However, there are dependencies defined on the attributes which can be violated, making it also a data cleaning problem.

For example, on a real-world online shopping web site, two sellers (or sources), s_1 and s_2, provide information about three phone models: *(Huawei) Honor7*, *(Huawei) Mates*, and *(Sony) Z3*. For simplicity, three representative attributes are considered, i.e., Central Processing Unit model (CPU for short), number of cores (#Core), and frequency. In addition to the key dependencies (the phone models are the key), there are two other FDs between these attributes:

- $\varphi_1 \equiv CPU \rightarrow \#Core$ indicates that phones with the same CPU model must contain the same number of cores.
- $\varphi_2 \equiv CPU \rightarrow Frequency$ indicates that phones with the same CPU model must also have an equal frequency.

Claims and ground truths are listed in Tables 1 and 2, respectively (where Snapdragon is abbreviated to Dragon to save space), and wrongly claimed values are underlined.

In a traditional truth discovery framework, s_1 and s_2 are initialized with uniform weights. With a probability of 0.5, or even higher in some methods because s_1 provides more claims than s_2, the wrong value *4* will win out as *Honor7*'s core number, which is undesirable. Meanwhile, only s_1 claims on *Mates*, which will result in a low confidence, especially when the only claiming source is of low reliability. Possible results of the truth discovery

Table 1 Claim tuples of mobile phones.

sid	Model	CPU	#Core	Frequency (GHz)
s_1	Honor7	Kirin935	$\underline{4}$	$\underline{2.2}$
s_1	Mates	Kirin935	8	$\underline{2.2}$
s_1	Z3	$\underline{\text{Dragon808}}$	4	$\underline{2.7}$
s_2	Honor7	Kirin935	8	1.5+2.2
s_2	Z3	Dragon801	4	2.5
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$

Table 2 Ground truth tuples of mobile phones.

Model	CPU	#Core	Frequency (GHz)
Honor7	Kirin935	8	1.5+2.2
Mates	Kirin935	8	1.5+2.2
Z3	Dragon801	4	2.5
$\cdots$	$\cdots$	$\cdots$	$\cdots$

methods are shown in Table 3, where the first two tuples violate φ_1. To resolve this inconsistency, one can either change *4* in the first tuple into *8*, or change *8* in the second tuple into *4*, which introduces a tie and may return a wrong answer. Furthermore, the tuple for *Z3* contains two wrong values and does not violate the dependencies. Therefore, the wrong values will not be detected, even though s_2 does claim correct values on $Z3$.

This example illustrates the problems faced when resolving data inconsistency and will be employed as a running example in the following sections to illustrate the method proposed in this paper.

4 Methodology

4.1 Preliminaries and problem definitions

Several important terms are introduced in this section, and the truth discovery problem in relational data is then defined.

Definition 1 Let $\mathcal{O}$ be a set of *objects*, where each object is a thing or a type of things. $\mathcal{A}$ is a set of concerned *attributes* of the objects. An *entry* is an object-attribute pair, and each entry should be identified by an identifier (*key* or *eid*) and should be assigned with a *value* as the *truth*. Values about the same object can be seen as a relational *tuple*.

In Table 2 for example, *Honor7*, *Mates*, and *Z3* are objects (phone models), and *CPU*, *#Core*, and *Frequency* are attributes. The entry corresponding to *Mates* and *CPU*, which can be denoted as *Mates[CPU]* or *Mates.CPU*, is assigned with a value *Kirin935* as the truth, meaning that *Mates* is equipped with the CPU *Kirin935*. There are 3 corresponding tuples describing the objects.

Table 3 Possible truth discovery results.

Model	CPU	#Core	Frequency (GHz)
Honor7	Kirin935	$\underline{4}$	$\underline{2.2}$
Mates	Kirin935	8	$\underline{2.2}$
Z3	$\underline{\text{Dragon808}}$	4	$\underline{2.7}$
$\cdots$	$\cdots$	$\cdots$	$\cdots$

Definition 2 Let $\mathcal{S}$ be a set of sources (i.e., a set of data providers) and $\mathcal{C}$ a set of *claims*, each of which is triple $\langle s,k,v\rangle$, meaning that the source, s, claims a value, v, on entry with key, k. Sources often provide claims in the form of tuples, and tuples from all sources are organized as a claim table, $\mathcal{CT}$. Different sources have differing degrees of reliability, which are reflected in the differing *sources*.

For example, in Table 1 there are two sources, s_1 and s_2, providing five claim-tuples for the three objects. Note that sources can conflict with each other, for instance, s_1 and s_2 claim that the frequencies of *Z3* are 2.7 GHz and 2.5 GHz, respectively. However, only one of the conflicting values actually agrees with the ground truth.

Definition 3 (Refer to Ref. [15]) An FD φ is of the form

$$[A_1,A_2,\cdots,A_l]\rightarrow[B],$$

where $A_i\in\mathcal{A}$ and $B\in\mathcal{A}$. The brackets can be omitted when there is only one attribute on the left (resp. right). The semantic of φ is that for any two tuples, t_1 and t_2, one of the following conditions should be satisfied,

- $t_1[A_i]\neq t_2[A_i]$ for some $i=1,2,\cdots,l$ or
- $t_1[B]=t_2[B]$.

The instinctive understanding is that if any two tuples are equal on the left-hand side attribute of φ, they should also be equal on the right-hand side attribute. In this paper, for any tuple, $t\in\mathcal{CT}$, we say that entry $t[B]$ is *covered* by φ if $t[A_i]\neq null$ for $i=1,2,\cdots,l$. In the running example, there are two FDs $CPU\rightarrow\#Core$, $CPU\rightarrow Frequency$. In Table 1, all entries relating to the attributes $\#Core$ and $Frequency$ are covered because there are no $null$ values. The first two tuples together violate $CPU\rightarrow\#Core$, because they are equal for CPU but differ with respect to the number of cores.

Definition 4 (Refer to Ref. [16]) A CFD ϕ is of the form

$$[A_1=v_1,A_2=v_2,\cdots,A_l=v_l]\rightarrow[B=v_0],$$

where each of $A_i,B\in\mathcal{A}$, and v_i is either a constant value or an unnamed variable '_'. The semantic of ϕ is that for any two tuples, t_1 and t_2, one of the following conditions should be satisfied,

- $t_1[A_i]\overset{v_i}{\neq}t_2[A_i]$ for some $i=1,2,\cdots,l$ or
- $t_1[B]\overset{v_0}{=}t_2[B]$,

where $t_1[A] \overset{v}{=} t_2[A]$ means that t_1 and t_2 are equal for A with a value, v, i.e., $t_1[A] = t_2[A]$ if v = '_', or $t_1[A] = t_2[A] = v$ if v is a constant. On the other hand, if $t_1[A] \overset{v}{=} t_2[A]$ does not hold, t_1 and t_2 are not equal for A with a value v, which is denoted by $t_1[A] \overset{v}{\neq} t_2[A]$.

The instinctive understanding is that if any two tuples are conditionally equal on the left-hand side attribute of ϕ, they should also be equal on the right-hand side attribute, and they should have corresponding values. For any tuple, t, that matches the left-hand side of ϕ, in this paper we say that entry $t[A]$ is *covered* by ϕ.

For example, if CFD $[Model = Mates] \to [CPU = Kirin935]$, it means that the phone model, *Mates*, is equipped with *Kirin935*, which is satisfied by Table 1, and the entry $Mates[CPU]$ is covered. In another example, the CFD $[Model = Z3, CPU = \text{'\_'}] \to [Frequency = \text{'\_'}]$ means that the *frequency* of a phone, *Z3*, depends on its *CPU*.

In this paper, all (conditional) FDs are satisfiable and non-redundant[16, 17, 52] in default.

The truth discovery task relating to inconsistent relational data is then formally defined.

Definition 5 Given claim table, $\mathcal{CT}$, about attributes in $mathcalA$ for objects in $\mathcal{O}$ provided by sources in $\mathcal{S}$, along with a set Σ of FDs and CFDs, the goal is to determine the most trustworthy values for each object, o, while satisfying all the dependencies in Σ.

4.2 Schema-Decomposing based Truth Discovery (SDTD) framework

The straightforward method for determining truths relating to each entry, and for cleaning the results according to Σ, has been shown in the previous problems. In this paper, an SDTD framework is considered, where the problem becomes a classical truth discovery problem (except that the initial weights and truth values require more elaborate calculations).

This method requires an initial pre-processing step. Each claim-tuple is decomposed into several source-key-value triples, using certain decomposing rules with respect to the dependencies. The aim of this is to consider the left-hand values with respect to a dependency of a virtual object and the right-hand value as the claimed value. After decomposition, a set of source-key-value triples is obtained, on which a truth discovery algorithm is conducted. In the claims obtained, it is possible for one source to claim different values for the same (virtual) object; therefore, the initial source weights and truth values in each iteration need to be carefully determined to provide highly accurate results. After conducting the truth discovery process, a post-processing step joins the key-value pairs into relational tuples, which directly satisfies the dependencies in most real-world situations.

We now discuss several key points within the SDTD framework in detail.

4.2.1 Decomposing rules

The functional dependency $\varphi \equiv X \to A$ means that tuples with the same values for X should be equal for A, where X is a set of attributes. This semantic happens to coincide with truth discovery settings, where the same object should hold the same value. In this respect, we consider extracting source-key-value triples from the claim-tuples according to the FDs using Rule 1:

- **Rule 1**. For each functional dependency, φ, decompose $\mathcal{CT}$ to obtain a new table, T_φ, using the projection operation, as

$$T_\varphi = \pi_{sid,X,A}(\mathcal{CT}),$$

where X is the set of attributes on the left-hand side of φ, and for A on the right-hand side.

For a CFD, $\phi \equiv [B_1 = v_1, B_2 = v_2, \cdots, B_l = v_l] \to [A = v_0]$, all tuples that match the embedded constant values should hold the same value for B whenever they are equal on the left-hand side attributes. Similar to Rule 1, source-key-value triples from the matched tuples according to the CFDs are extracted using Rule 2.

- **Rule 2**. For each conditional functional dependency $\phi \equiv [B_1 = v_1, B_2 = v_2, \cdots, B_l = v_l] \to [A = v_0]$, decompose $\mathcal{CT}$ to obtain a new table, T_ϕ, using the selection and projection operations,

$$T_\phi = \pi_{sid,B_1,\cdots,,B_l,A}(\sigma_{B_{c1}=v_{c1},\cdots,B_{cn}=v_{cn}}(\mathcal{CT})),$$

where $B_{c1}, \cdots, B_{cn}$ are attributes embedded with constant values, $v_{c1}, \cdots, v_{cn}$, respectively.

In addition, if ϕ in Rule 2 is embedded with a constant value for the right-hand side attribute, A, it is claimed to be a correct key-value pair. Rule 3 is employed to represent the same semantic as follows:

- **Rule 3**. For each CFD $\phi \equiv [B_1 = v_1, B_2 = v_2, \cdots, B_l = v_l] \to [A = v_0]$, where v_0 is a constant value, construct a single tuple, t_ϕ, as shown in Table 4, and insert it into T_ϕ, where $B_{c1}, \cdots, B_{cm}$ are attributes in ϕ bound with constant values $v_{c1}, \cdots, v_{cm}$, and $ssid$ is a virtual *super source*, which always provides true

Table 4 Single tuple constructed from ϕ.

Sid	B_{c1}	B_{c2}	$\cdots$	B_{cm}	A
ssid	v_{c1}	v_{c2}	$\cdots$	v_{cm}	v_0

values.

As some entries may not be covered by any dependency, Rule 4 is introduced to deal with these as follows:

- **Rule 4**. If a source, s, claims value, v, on an uncovered entry, $o[A]$, insert a claim-tuple (s, o.'A', v) into the table $T_{rest}(sid, key, value)$.

All the four rules generate tuples that consist of a source, s, and a key, k, which is composed of the left-hand side attribute value(s), and a value, v; and the tuples can be seen as claims in the natural form of $\langle sid, key, value\rangle$.

For example, if Table 1 is taken as the input claim table $\mathcal{CT}$, along with three dependencies $\varphi \equiv [CPU] \rightarrow [\#Core]$, $\phi_1 \equiv [CPU = Kirin935, \#Core = \text{'\_'}] \rightarrow [Frequency = \text{'\_'}]$, and $\phi_2 \equiv [CPU = Dragon801] \rightarrow [Core = 8]$, the tables generated according to the decomposing rules are shown (from Table 5 to Table 8).

All tuples in these tables can be seen as $\langle sid, key, value\rangle$ triples: the $sids$ can be obtained directly; the $value$ corresponds to the last attribute, which is usually the right-hand side attribute; and all the other attributes are combined as $keys$. To identify which attributes do the values that constitute the key come from, it is necessary to consider the attribute names as part of the key. For example, the key corresponding to the first tuple in Table 6 would be "CPU.Kirin935; Frequency.2.2GHz".

Table 5 T_φ generated according to Rule 1.

sid	CPU	#Core
s_1	Kirin935	4
s_1	Kirin935	8
s_1	Dragon808	4
s_2	Kirin935	8
s_2	Dragon801	4
...	...	...

Table 6 T_{ϕ_1} generated according to Rule 2.

sid	CPU	#Core	Frequency (GHz)
s_1	Kirin935	4	2.2
s_1	Kirin935	8	2.2
s_2	Kirin935	8	1.5+2.2
...	...	...	...

Table 7 T_{ϕ_2} generated according to Rules 2 and 3.

sid	CPU	#Core
s_2	Dragon801	4
$ssid$	Dragon801	8
...	...	...

Table 8 T_{rest} generated according to Rule 4.

sid	Key	Value
s_1	Z3.Frequency	2.7 Hz
s_1	Honor7.CPU	Kirin935
s_1	Mates.CPU	Kirin935
s_1	Z3.CPU	Dragon808
s_2	Honor7.CPU	Kirin935
s_2	Z3.CPU	Dragon801
s_2	Z3.Frequency	2.5 Hz
...	...	...

4.2.2 Determining source weights

The basic idea with all truth discovery methods is that sources have different reliability levels, and that higher weights should be assigned to sources that are more reliable. In this paper, the probability of a source, s, of giving correct values is estimated as its weight.

At the beginning of the iterations, almost all the existing truth discovery methods start with uniform source weights. In our setting, a single source may provide self-conflicting data, which indicates the source has low reliability. With the self-conflictions, source weights can be initialized more elaborately.

For example in Table 5, s_1 claims two different values, 4 and 8, for the core number of *Kirin935*, and no more than one of these can be correct. The upper bound of number correct claims provided by source s over key k is

$$u_{s,k} = \begin{cases} N_{s,k,v}, & \text{if } \exists v(N_{ssid,k,v} > 0); \\ \max\limits_{v}(N_{s,k,v}), & \text{else} \end{cases} \tag{1}$$

where $N_{s,k,v}$ is the number of claims provided by s for k with a value v. Equation (1) means that if the super source claims that v is the absolute true value of k, the upper bound represents the number of correct claims. However, if the super source provides no truth value for k, the self-conflicting value that has the most votes is assumed to be true. The initial weight of source s can be estimated according to the upper bounds as

$$w_s^{(0)} = \frac{\sum_k u_{s,k}}{\sum_{k,v}(N_{s,k,v})} \tag{2}$$

The denominator is the total claims provided by s and the numerator is the maximum possible number of correct ones. Obviously, all weights are in $[0,1]$, and sources providing less-conflicting values gain higher upper bounds, as well as higher weights. In particular, if a source provides no self-conflictions and always agrees with the super source, it will be initialized with the highest weight, 1.0.

The situation improves during iterations because estimated truths become available. If the estimated value of the key, k, in the i-th iteration is denoted by $v_k^{(i)}$, the source weights in the i-th iteration can be then naturally estimated by

$$w_s^{(i)} = \frac{\sum_k N_{s,k,v_k^{(i)}}}{\sum_{k,v}(N_{s,k,v})} \tag{3}$$

i.e., the probability of source s providing correct claims.

4.2.3 Determining truths

With the source weights (or probabilities of providing correct values) previously estimated, a probabilistic model is analyzed to determine the truths. By providing the claim table, $\mathcal{CT}$, the true values can be estimated according to conditional probabilities.

For a single claim $\langle s,k,v\rangle$ in $\mathcal{CT}$, the probability of v being correct, denoted as $P_{k,v}$, can be directly assigned with the probability of s providing correct values, i.e., w_s.

However, in our settings, it is possible to have multiple conflicting claims from the same source over the same key, and if a claim is involved in a conflict, its probability of being correct should be reduced. If source, s, provides $N_{s,k}$ claims over key k, among which $N_{s,k,v}$ claims are with value v, k is composed of m attributes, as shown in Table 9. In the condition where s provides a correct value, the probability of the claim $\langle s,k,v\rangle$ being correct is naturally estimated by voting, i.e.,

$$P\{\langle s,k,v\rangle \text{ is correct}|s \text{ gives a true value on } k\} = N_{s,k,v}/N_{s,k} \tag{4}$$

under the intuition that the value that has more votes is more likely to be correct.

In the condition that the claims giving v are incorrect, it is possible for v or $a_1,\cdots,a_m$ to be wrong. It is assumed that all the $m+1$ values have an equal probability of being incorrect, therefore, it can be inferred approximately that

$$P\{a_i \text{ in } k \text{ is correct}|\text{claim } \langle s,k,v\rangle \text{ is incorrect}\} = P\{v \text{ is correct}|\text{claim } \langle s,k,v\rangle \text{ is incorrect}\} = m/(m+1) \tag{5}$$

With Eq. (4) and Eq. (5), when values claimed by s over k conflict with each other, the probability of each value involved being correct should be multiplied by a discount factor,

$$df_{s,k,v} = \frac{N_{s,k,v}}{N_{s,k}} + \frac{N_{s,k}-N_{s,k,v}}{N_{s,k}} \times \frac{m}{m+1} \tag{6}$$

where m is the number of attributes in key k. Note that the discount factor depends on the claim table, $\mathcal{CT}$, and the set Σ of dependencies only, which means it is sufficient to calculate it just once before the iterations. The discount factor is of the largest value, 1, when s provides no conflicting values for key, k. However, if there are conflicting values, the discount factor is strictly less than 1. It is worth noting that when the super-source claims a value on k, the discount factor can be calculated more accurately (however, the derivation is similar and is omitted here).

When an entry, e, is involved in more than one conflict, its probability of being correct should be discounted multiple times as

$$P_{s,k,v}^{(i+1)} = w_s^{(i)} \times \prod_{\text{each } \langle s,k',v'\rangle \text{ involving } e} df_{s,k',v'} \tag{7}$$

where k is the key identifying e. Equation (7) estimates the probability of v being correct in the $(i+1)$-th iteration.

Consider key, k, and candidate values $v_1,v_2,\cdots,v_c$, if it is assumed that there is no relationship between sources as

$$P\{\mathcal{CT} \text{ is claimed}|v_k^*=v\} \propto \prod_{\langle s,k,v\rangle} P_{s,k,v} \times \prod_{\langle s,k,v_j\rangle, v_j\neq v}(1-P_{s,k,v_j}) \propto \prod_{\langle s,k,v\rangle}\frac{P_{s,k,v}}{1-P_{s,k,v}} \tag{8}$$

where v_k^* denotes the true value of key k, and the conditional probability is abbreviated as $P\{\mathcal{CT}|v_k^*=v\}$ for convenience. According to Bayesian formula, the posteriori probabilities can be derived as

$$P\{v_k^*=v|\mathcal{CT}\} = \frac{P\{\mathcal{CT}|v_k^*=v\}\times P\{v_k^*=v\}}{\sum_j(P\{\mathcal{CT}|v_k^*=v_j\}\times P\{v_k^*=v_j\})} \tag{9}$$

We assume that no previous knowledge is available, and priori probability equals for all candidate values, therefore,

$$P\{v_k^*=v|\mathcal{CT}\} = \frac{P\{\mathcal{CT}|v_k^*=v\}}{\sum_j P\{\mathcal{CT}|v_k^*=v_j\}} \propto$$
$$P\{\mathcal{CT}|v_k^*=v\} \propto \prod_{\langle s,k,v\rangle}\frac{P_{s,k,v}}{(1-P_{s,k,v})} \tag{10}$$

Table 9 Claims provided by s over k.

Claim id	Sid	Key			Value
c_1	s	a_1	$\cdots$	a_m	v
$\cdots$	s	a_1	$\cdots$	a_m	$\cdots$
$c_{N_{s,k,v}}$	s	a_1	$\cdots$	a_m	v
$c_{N_{s,k,v}}+1$	s	a_1	$\cdots$	a_m	v'
$\cdots$	s	a_1	$\cdots$	a_m	$\cdots$
$c_{N_{s,k}}$	s	a_1	$\cdots$	a_m	$\cdots$

The value with the highest posterior probability is thus selected as the truth,

$$v_k^{(i)} = \underset{v}{\text{argmax}} \prod_{\langle s,k,v \rangle} \frac{P_{s,k,v}}{(1-P_{s,k,v})} \tag{11}$$

or equally

$$v_k^{(i)} = \underset{v}{\text{argmax}} \sum_{\langle s,k,v \rangle} (\ln P_{s,k,v}^{(i)} - \ln(1-P_{s,k,v}^{(i)})) \tag{12}$$

4.2.4 Composition process

When truth discovery iterations are terminated, a relational table, T_c, is constructed from all truth pairs of the form $\langle k,v \rangle$.

The value for each tuple on attribute A can be filled using the estimated set T^* of truths, along with the corresponding object, o. In the decomposing process, every entry value is guaranteed to be covered by a rule. If it is covered by Rule 4, then the entry value can be directly set as the discovered v_k^* where $k = o.A$, and if it is covered by Rules 1−3, then all corresponding left-hand values should be filled in advance. The composition process is illustrated as the *Fill procedure*, as shown in Algorithm 1.

In our settings, it is assumed that each attribute is in the right-hand side of one dependency at most, and that there are no cycles in the depending relationships that can be satisfied in most real-world situations. When these assumptions hold, each entry's value can be determined by exactly one key, and no conflicts occur in the composition process. Other general cases will be considered in future work.

4.2.5 Algorithm flow

Several key points have already been described. This subsection summarizes the overall flow for the proposed SDTD method, shown in Algorithm 2, and this comprises the following three main steps,

Step 1: Preprocess. SDTD first computes discount factors for entries according to Eq. (6) (line 1) and then decomposes relational tuples into triple-formed claims according to the four decomposing rules (lines 2−11).

Step 2: Truth discovery iterations. The source weights are initialized according to Eqs. (1) and (2) (line 12). The true values are then iteratively estimated and source weights are updated until a stopping criterion is met (lines 13−19).

Step 3: Postprocess. Truths discovered in key-value form are composed back into relational tuple form, and T_c is created (lines 20−24).

The preprocess and postprocess deal with the claim table, $\mathcal{CT}$, and the truth table, T_c, only once, respectively. However, it is obvious that there are complexities $O(|\mathcal{CT}| \times m)$ and $O(|\mathcal{O}| \times m)$, where m is the number of attributes. In the truth discovery iterations, there are $|\mathcal{CT}| \times m$ entries, and the time complexity is $O(|\mathcal{CT}| \times m \times k)$, where k is the number of iterations. Thus the overall complexity is $O(|\mathcal{CT}| \times m \times k)$, which is linear with the number of claim tuples, indicating a good scalability.

4.2.6 Practical issues

Several practical issues should be considered in the SDTD framework.

Null values are very common in the database community due to incompleteness. In the decomposing process, if a null value occurs in the right-hand side of a

Algorithm 1 $\boldsymbol{Fill(T_c, o, A, \Sigma, T^*)}$

```
1: if $o[A]$ is already filled then
2:     return
3: if There is a dependency $X \to A \in \Sigma$ then
4:     for Each attribute $B \in X$ do
5:         $Fill(T_c, o, B, \Sigma, T^*)$
6:     Compose key $k$ by values on $X$ and $A$
7: else
8:     Compose key $k$ by $o$ and the attribute $A$
9: Fill $o[A]$ in $T_c$ with value $v_k^*$, which can be found in $T^*$
10: return
```

Algorithm 2 SDTD

```
Input: A relational claim table $\mathcal{CT}$, a set of FDs $\Sigma$.
Output: The cleaned relational table $T_c$ satisfying $\Sigma$.
1: Calculate the discount factor for each entry according to Eq. (6)
2: for Each dependency $\phi \in \Sigma$ do
3:     for Each tuple $t \in \mathcal{TC}$ do
4:         Generate claim $c$ according to Rule 1 or Rule 2 respectively with $\phi$ on $t$
5:         Add $c$ into claim set $\mathcal{C}$
6:     if $\phi$ is a constant CFD then
7:         Generate claim $c$ according to Rule 3 with $\phi$ on $t$
8:         Add $c$ into claim set $\mathcal{C}$
9: for Each uncovered entry do
10:    Create claim $c$ according to Rule 4 on $t$
11:    Add $c$ into claim set $\mathcal{C}$
12: Initialize source weights according to Eq. (1) and Eq. (2)
13: repeat
14:    for Each claim $\langle s,k,v \rangle \in \mathcal{C}$ do
15:        Compute its relative probability of being true according to Eq. (7)
16:    for Each $k$ do
17:        Update the true value of $k$ in $T^*$ according to Eq. (12)
18:    Update source weights using the estimated values according to Eq. (3).
19: until A stopping criterion is met.
20: Create a table $T_c$ of tuples for each object, with other attributes unfilled.
21: for Each tuple representing $o$ do
22:    for Each attribute $A$ do
23:        $Fill(T_c, o, A, \Sigma, T^*)$
24: return $T_c$;
```

dependency, then the generated triple should be discarded because it provides no valuable information. If a null value occurs in the left-hand side of a dependency, the generated triple should also be discarded, because it creates an uncertain key-value that causes problems in the iterating process. In the second case, the right-hand side value becomes uncovered and should be dealt with using Rule 4, which then ensures complete answers in the final result.

Only categorial attributes have been discussed in this paper; however, numeric values can be easily integrated into our framework. A numeric valued attribute seldom occurs in the left-hand side of a dependency, (if it does, it can be treated as a categorial attribute). In the truth discover iterations, the source weights can be calculated more carefully if numeric values are available, and true numeric values can be computed as weighted averages among multiple sources.

In Eq. (12), it is very possible that $P^{(i)}_{s,k,v}$ equals 0 or 1, and that it causes invalid numbers. To solve this problem, a bounded interval is imposed on the probability,

$$P^{(i)}_{s,k,v} := \max(P^{(i)}_{s,k,v}, \epsilon),$$

and

$$P^{(i)}_{s,k,v} := \min(P^{(i)}_{s,k,v}, 1-\epsilon),$$

where ϵ is a positive value close to zero, which is set as 0.01 in our experiments.

Finally, when calculating the discount factors and estimating the values, if the super source claims a value on the key, the calculations should be adjusted accordingly; as this is intuitive, formal discussions are omitted here.

5 Experiment

This section analyzes the performance of our method using two real world data sets and synthetical data sets. The results show that when data cleaning and truth discovery tasks are simultaneously considered, a superb data quality can be obtained when dealing with relational data, which violates dependencies provided by multiple sources.

5.1 Experiment setup

All experiments were conducted using JAVA and performed on a Dell desktop with an Octa-Core Intel i7 3.60 GHz CPU, 8 GB of memory, a 1 TB SATA disk, and with Windows 7 as the underlying operating system.

5.1.1 Data sets

Experiments were conducted on both real-world data and synthetic data.

The real-world data set contains two subsets: *Notebooks* and *Phones*, which are obtained from one of the most popular shopping web sites. Many shops (sources) sell various products (objects) and list their corresponding specifications (attributes) to customers. The statistical information of the two subsets is listed in Table 10. For simplicity, only several attributes are provided for the two data sets. The *Notebooks* data set contains three attributes: *scr_res* (screen resolution), *scr_rate* (aspect ratio), and *scr_type* (screen type: wide or standard), and there are two dependencies between the attributes: $scr\_res \to scr\_rate$ and $scr\_rate \to scr\_type$. Meta data of the *Phone* data set can be found in Section 3. To ensure correctness, all ground truths were collected manually from corresponding official web sites; however, although we made our best efforts to collect ground truths that were as complete as possible, only a small portion of these were available, as shown in Table 10. However, the data provided are suitable for use in an accuracy evaluation.

In the synthetic data sets, 500 sources provided claims on 10 000 (or ranging from 10 000 to 100 000 in the scalability test) objects. For each object, *o*, five sources (on average) were randomly selected to claim on *o*; therefore, there were a total of approximately 50 000 claim-tuples (ranging from 50 000 to 500 000). The claiming inaccuracy (or error rate) of each source was a random value of a well-known distribution, $Beta(\alpha, \beta)$, because it is the most common that is continuous and defined with an interval [0, 1]. By varying α and β, expectation of the source inaccuracy ranges from 1% to 10%.

5.1.2 Baseline methods

The proposed method is compared with two latest algorithms, *CRH*[13] and *CATD*[14], as well as the simple *VOTE*, where the value with the highest number of votes is regarded as the truth. Because these three methods only enabled truth discovery, a cleaning process[19] was appended to them, and three new methods, *CRH+Clean*, *CATD+Clean*, and *VOTE+Clean*, were obtained.

For all iteration-based methods, a global threshold (defaulted 3) of the number of iterations was used for consideration of fairness.

Table 10 Real world data sets statistics.

Data set	Number of Sources	Objects	Claim tuples	Attributes	Ground truth
Notebooks	12 092	5696	60 151	3	2494
Phones	11 187	1595	37 053	3	1337

5.2 Assumptions validation

In this subsection, several assumptions that inspired our study are validated on two real world data sets.

5.2.1 Self-conflictions of sources

In our settings, an important assumption is that tuples from a single data source may violate some dependencies; this is validated in Fig. 1. Each data source is represented by a point, where the x-coordinate is the number of claims provided, and the y-coordinate is the number of conflicting values among the claims. Both of the real world data sets contained self-conflictions, and a source was more likely to contain self-conflictions when it provided more claims. Therefore, by obtaining the self-conflicting information, initial weights of the sources could be assigned more wisely.

5.2.2 Tie phenomenon

The number of ties is significant with respect to the uniform source-weight initialization, because the first iteration is a simple voting process. Ties play a very unstable role in truth discovery because more than one value has an equal probability of being selected as the true value. In Fig. 2, the tie phenomenon is illustrated more intuitively. Each point stands for an entry, where the x-coordinate is the number of candidate values of the entry, and the y-coordinate is the number of ties. The tie phenomenon is obvious, particularly in the *Phones* data set.

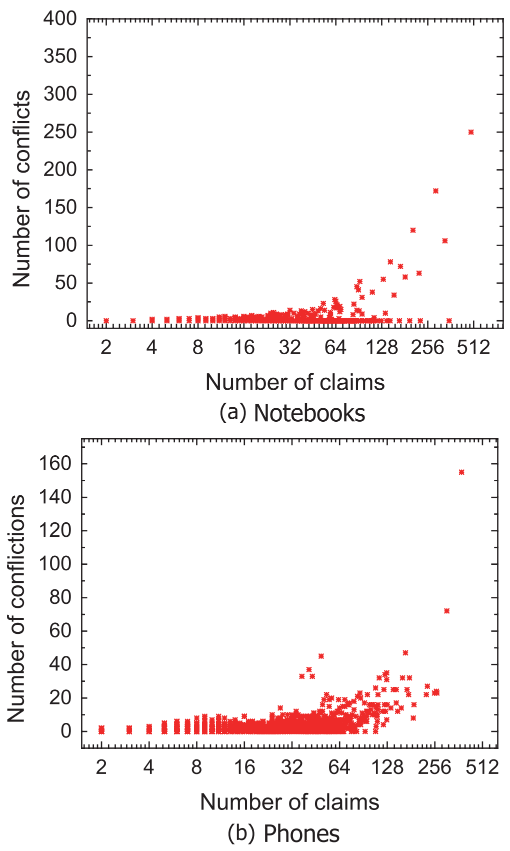


Fig. 1 Relationship between numbers of source claims and conflictions in real world data sets.

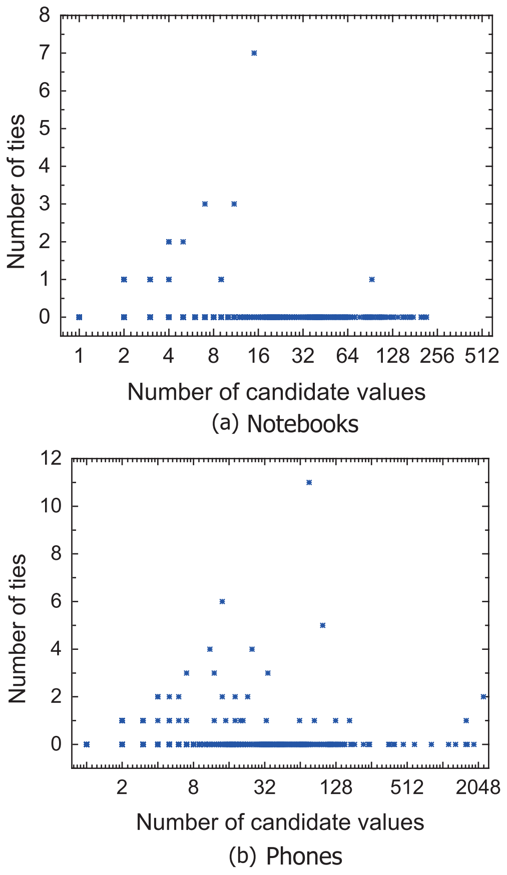


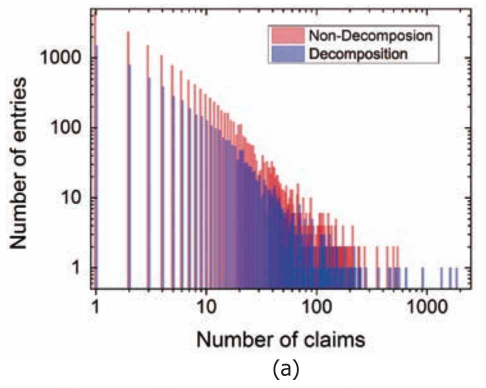
Fig. 2 Relationship between numbers of candidate values of the entries and ties with real world data sets.

5.2.3 Claim numbers before and after decomposition

This study has discussed that when very few sources provide claims on an entry, the discovered truth has a low confidence. Our schema-decomposing based framework can solve this problem to some extent. The number of claims made for each entry before and after decomposition can be counted, and Fig. 3 shows the distribution of the number of claims. It can also be seen that most entries are claimed by very few sources. By decomposition, our method decreases the number of entries that provide few claims by about 75%, and some entries with a larger number of claims are created (note that this is plotted using logarithmic coordinates).

5.3 Results

This subsection presents experimental results relating to effectiveness and efficiency.



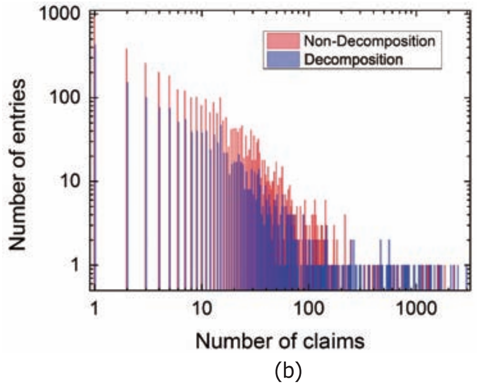
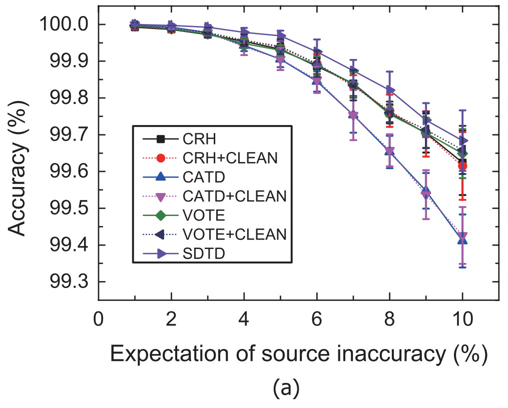


Fig. 3 Distribution of the number of claims of entries on real world data sets: (a) Notebooks and (b) Phones.



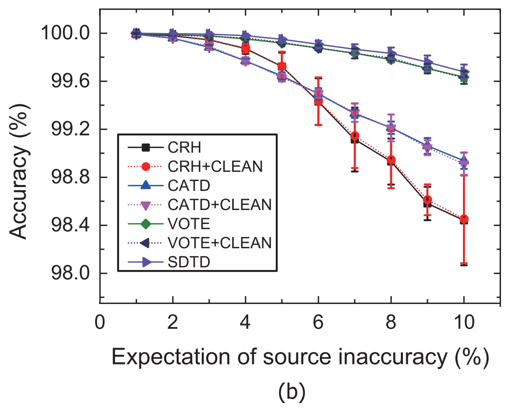


Fig. 4 Accuracy comparison when varying source reliability with (a) $\alpha+\beta=5$ and (b) $\alpha+\beta=20$.

5.3.1 Accuracy

The accuracy of all methods used with real world data sets is listed in Table 11. Results show that methods that did not employ a cleaning process performed badly, particularly on Notebooks, whereas the cleaning process significantly improved the accuracy to a certain extent. Of all the methods evaluated, SDTD performs the best with both Notebooks and Phone data sets.

Accuracy was evaluated for synthetic data sets, using different expectations of source-inaccuracy. To obtain results with a higher confidence, each test was conducted on synthetic data sets for 10 times, and the average value was used as the final result. From Fig. 4, it is evident that the SDTD performs well, even when the sources have different degrees of reliability.

5.3.2 Convergence rates

One of the important features of a good truth discovery method is its convergence rate, and this was evaluated here using real world data sets. Table 12 lists the relative changes of the different methods in each iteration, and it can be seen that CATD and SDTD converge fast while CRH keeps changing (because it gives two different sets of results in turn).

5.3.3 Scalability

Section 4.2 shows that the running-time of the proposed SDTD framework is linear with respect to the number of

Table 11 Accuracies using real world data sets.

Method	Notebooks	Phones
crh	0.833 600 642	0.715 781 601
crh+clean	0.936 246 993	0.730 740 464
catd	0.834 402 566	0.728 496 634
catd+clean	0.936 246 993	0.753 926 702
vote	0.836 407 378	0.725 504 862
vote+clean	0.936 647 955	0.750 186 986
sdtd	**0.957 898 957**	**0.766 641 735**

Table 12 Relative changes in each iteration.

Iter	Notebooks			Phones		
	crh	catd	sdtd	crh	catd	sdtd
2	1211	201	30	309	79	32
3	1259	16	1	319	1	1
4	1289	0	0	281	0	1
5	1297	0	0	281	0	0
6	1297	0	0	282	0	0
7	1295	0	0	282	0	0
8	1296	0	0	282	0	0

claim tuples. Its scalability is validated by varying the number of objects (doing so caused a linear increase in the number of claims). The average running times are shown to be scattered in Fig. 5, which reflects 10 running times for each scale with the synthetic data sets. From this, it is obvious that the data provide a good fit with linear regression (the red line), with a Pearson Correlation of 0.9894.

6 Conclusion

This paper studies the truth discovery problem with relational data and (conditional) FDs. There are several problems inherent in the use of both truth discovery and data cleaning, and a framework is established that considers both methods. A schema-decomposing based method is then proposed that discovers true values among sources with different reliability degrees, and the output truths directly satisfy the given dependencies. Experiments using both real and synthetic data sets validate the effectiveness and efficiency of the proposed method.

In future work, more sophisticated situations will be considered to include, but will not be limited to, the use of a greater number of general dependency cases (in this paper, no cycle between dependencies is assumed, and every attribute is covered by no more than one dependency). In addition, differential dependencies will be considered, as will the use of more data quality constraining types, such as editing rules and fixing rules.

Acknowledgment

This paper was partially supported by the Key Research and Development Plan of National Ministry of Science and Technology (No. 2016YFB1000703), the Key Program of the National Natural Science Foundation of China (Nos. 61190115, 61472099, 61632010, and U1509216), National Sci-Tech Support Plan (No. 2015BAH10F01), the Scientific Research Foundation for the Returned Overseas Chinese Scholars of Heilongjiang Province (No. LC2016026), and MOE-Microsoft Key Laboratory of Natural Language Processing and Speech, Harbin Institute of Technology.

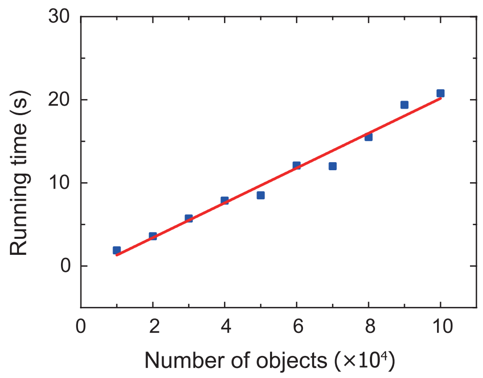


Fig. 5 Relationship between running time and data size.

References

[1] W. Fan and F. Geerts, Cleaning data with conditional dependencies, in *Foundations of Data Quality Management*, M. T. Özsu, ed. San Rafael, CA, USA: Morgan & Claypool Publishers, 2012, pp. 39–86.

[2] X. Li, X. L. Dong, K. Lyons, W. Meng, and D. Srivastava, Truth finding on the deep web: Is the problem solved? in *Proc. 39th Int. Conf. Very Large Data Bases*, Riva del Garda, Italy, 2013, pp. 97–108.

[3] X. L. Dong, L. B. Equille, and D. Srivastava, Data fusion: Resolving conflicts from multiple sources, in *Handbook of Data Quality*, S. Sadiq, ed. Springer, 2013, pp. 293–318.

[4] X. Yin, J. Han, and P. S. Yu, Truth discovery with multiple conflicting information providers on the web, in *Proc. 13th Int. Conf. Knowledge Discovery and Data Mining*, San Jose, CA, USA, 2007, pp. 1048–1052.

[5] X. L. Dong, L. B. Equille, and D. Srivastava, Integrating conflicting data: The role of source dependence, in *Proc. 35th Int. Conf. Very Large Data Bases*, Lyon, France, 2009, pp. 550–561.

[6] A. Galland, S. Abiteboul, A. Marian, and P. Senellart, Corroborating information from disagreeing views, in *Proc. 3rd Int. Conf. Web Search and Web Data Mining*, New York, NY, USA, 2010, pp. 131–140.

[7] J. Pasternack and D. Roth, Knowing what to believe (when you already know something), in *Proc. 23rd Int. Conf. Computational Linguistics*, Beijing, China, 2010, pp. 877–885.

[8] X. Yin and W. Tan, Semi-supervised truth discovery, in *Proc. 20th Int. World Wide Web Conf.*, Hyderabad, India, 2011, pp. 217–226.

[9] B. Zhao, B. I. P. Rubinstein, J. Gemmell, and J. Han, A Bayesian approach to discovering truth from conflicting sources for data integration, in *Proc. 38th Int. Conf. Very Large Data Bases*, Istanbul, Turkey, 2012, pp. 550–561.

[10] B. Zhao and J. Han, A probabilistic model for estimating real-valued truth from conflicting sources, presented at the 10th Int. Workshop on Quality in Databases, Istanbul, Turkey, 2012.

[11] D. Wang, L. Kaplan, H. Le, and T. Abdelzaher, On truth discovery in social sensing: A maximum likelihood estimation approach, in *Proc. 11th Int. Conf. Information Processing in Sensor Networks*, Beijing, China, 2012, pp. 233–244.

[12] J. Pasternack and D. Roth, Latent credibility analysis, in *Proc. 22nd Int. World Wide Web Conf.*, Rio de Janeiro, Brazil, 2013, pp. 1009–1020.

[13] Q. Li, Y. Li, J. Gao, B. Zhao, W. Fan, and J.

Han, Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation, in *Proc. Int. Conf. Management of Data*, Snowbird, UT, USA, 2014, pp. 1187–1198.

[14] Q. Li, Y. Li, J. Gao, L. Su, B. Zhao, M. Demirbas, W. Fan, and J. Han, A confidence-aware approach for truth discovery on long-tail data, in *Proc. 41st Int. Conf. Very Large Data Bases*, Kohala Coast, HI, USA, 2015, pp. 425–436.

[15] E. F. Code, Relational completeness of data base sublanguages, in *Data Base Systems, Courant Computer Science Symposia 6*, R. Rustin, ed. Upper Saddle River, NJ, USA: Prentice Hall, 1972, pp. 65–98.

[16] P. Bohannon, W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis, Conditional functional dependencies for data cleaning, in *Proc. 23rd Int. Conf. Data Engineering*, Istanbul, Turkey, 2007, pp. 746–755.

[17] W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis, Conditional functional dependencies for capturing data inconsistencies, *ACM Trans. Database Syst.,* vol. 33, no. 6, pp. 1–48, 2008.

[18] W. Fan, J. Li, N. Tang, and W. Yu, Incremental detection of inconsistencies in distributed data, *IEEE Trans. Knowl. Data Eng.,* vol. 26, no. 6, pp. 1367–1383, 2014.

[19] P. Bohannon, W. Fan, M. Flaster, and R. Rastogi, A cost-based model and effective heuristic for repairing constraints by value modification, in *Proc. Int. Conf. Management of Data*, Baltimore, MD, USA, 2005, pp. 143–154.

[20] J. Chomicki and J. Marcinkowski, Minimal-change integrity maintenance using tuple deletions, *Inf. Comput.,* vol. 197, no. 2005, pp. 90–121, 2005.

[21] J. Wijsen, Database repairing using updates, *ACM Trans. Database Syst.,* vol. 30, no. 3, pp. 722–768, 2005.

[22] S. Kolahi and L. V. S. Lakshmanan, On approximating optimum repairs for functional dependency violations, in *Proc. 12th Int. Conf. Database Theory*, St. Petersburg, Russia, 2009, pp. 53–62.

[23] G. Beskales, I. F. Ilyas, and L. Golab, Sampling the repairs of functional dependency violations under hard constraints, in *Proc. 36th Int. Conf. Very Large Data Bases*, Singapore, 2010, pp. 197–207.

[24] G. Beskales, I. F. Ilyas, L. Golab, and A. Galiullin, Sampling from repairs of conditional functional dependency violations, *VLDB J.,* vol. 23, no. 1, pp. 103–128, 2014.

[25] L. Bravo, W. Fan, F. Geerts, and S. Ma, Increasing the expressivity of conditional functional dependencies without extra complexity, in *Proc. 24th Int. Conf. Data Engineering*, Cancún, México, 2008, pp. 516–525.

[26] W. Chen, W. Fan, and S. Ma, Analyses and validation of conditional dependencies with built-in predicates, in *Proc. 20th Int. Conf. Database and Expert Systems Applications*, Linz, Austria, 2009, pp. 576–591.

[27] W. Fan, J. Li, S. Ma, N. Tang, and W. Yu, Towards certain fixes with editing rules and master data, *VLDB J.,* vol. 21, no. 2, pp. 213–238, 2012.

[28] J. Wang and N. Tang, Towards dependable data repairing with fixing rules, in *Proc. Int. Conf. Management of Data*, Snowbird, UT, USA, 2014, pp. 457–468.

[29] S. Song and L. Chen, Differential dependencies: Reasoning and discovery, *ACM Trans. Database Syst.,* vol. 36, no. 16, pp. 1–41, 2011.

[30] S. Song, L. Chen, and P. S. Yu, Comparable dependencies over heterogeneous data, *VLDB J.,* vol. 22, no. 2, pp. 253–274, 2013.

[31] Y. Li, J. Gao, C. Meng, Q. Li, L. Su, B. Zhao, W. Fan, and J. Han, A survey on truth discovery, *SIGKDD Explorations,* vol. 17, no. 2, pp. 1–16, 2015.

[32] D. Qiu, L. Barbosa, X. L. Dong, Y. Shen, and D. Srivastava, DEXTER: Large-scale discovery and extraction of product specifications on the web, in *Proc. 41st Int. Conf. Very Large Data Bases*, Kohala Coast, HI, USA, 2015, pp. 2194–2205.

[33] M. Yakout, A. K. Elmagarmid, J. Neville, and M. Ouzzani, GDR: A system for guided data repair, in *Proc. Int. Conf. Management of Data*, Indianapolis, IN, USA, 2010, pp. 1223–1226.

[34] M. Yakout, A. K. Elmagarmid, J. Neville, M. Ouzzani, and I. F. Ilyas, Guided data repair, in *Proc. 37th Int. Conf. Very Large Data Bases*, 2011, pp. 279–289.

[35] H. Xie, H. Wang, J. Li, and H. Gao, A data cleaning framework based on user feedback, in *Proc. 14th Int. Conf. Web-Age Information Management*, Beidaihe, China, 2013, pp. 514–520.

[36] J. He, E. Veltri, D. Santoro, G. Li, G. Mecca, P. Papotti, and N. Tang, Interactive and deterministic data cleaning, in *Proc. Int. Conf. Management of Data*, San Francisco, CA, USA, 2016, pp. 893–907.

[37] Z. Cai, Z. He, X. Guan, and Y. Li, Collective data-sanitization for preventing sensitive information inference attacks in social networks, *IEEE Trans. Depend. Secure.*, doi:10.1109/TDSC.2016.2613521.

[38] Z. He, Z. Cai, Y. Sun, Y. Li, and X. Cheng, Customized privacy preserving for inherent-data and latent-data, *Pers. Ubiquit. Comput.,* vol. 21, no. 1, pp. 43–54, 2017.

[39] D. Miao, Z. Cai, X. Liu, and J. Li, Functional dependency restricted insertion propagation, *Theoret. Comput. Sci.*, doi:10.1016/j.tcs.2017.03.043.

[40] D. Miao, Z. Cai, X. Liu, and J. Li, On the complexity of insertion propagation with functional dependency constraints, in *Proc. 22nd Int. Conf. Computing and Combinatorics*, Ho Chi Minh City, Vietnam, 2016, pp. 623–632.

[41] Z. Cai, M. Heydari, and G. Lin, Iterated local least squares microarray missing value imputation, *J. Bioinf. Comput.*

Biol., vol. 4, no. 4, pp. 935–958, 2006.

[42] X. Ding, H. Wang, Y. Gao, J. Li, and H. Gao, Efficient currency determination algorithms for dynamic data, *Tsinghua Sci. Technol.,* vol. 22, no. 3, pp. 227–242, 2017.

[43] X. Chu, I. F. Ilyas, S. Krishnan, and J. Wang, Data cleaning: Overview and emerging challenges, in *Proc. Int. Conf. Management of Data*, San Francisco, CA, USA, 2016, pp. 2201–2206.

[44] J. Pasternack and D. Roth, Making better informed trust decisions with generalized fact-finding, in *Proc. 22nd Int. Joint Conf. Artificial Intelligence*, Barcelona, Spain, 2011, pp. 2324–2329.

[45] M. Gupta, Y. Sun, and J. Han, Trust analysis with clustering, in *Proc. 20th Int. World Wide Web Conf.*, Hyderabad, India, 2011, pp. 53–54.

[46] D. Wang, T. F. Abdelzaher, H. Ahmadi, J. Pasternack, D. Roth, M. Gupta, J. Han, O. Fatemieh, H. K. Le, and C. C. Aggarwal, On bayesian interpretation of fact-finding in information networks, in *Proc. 14th Int. Conf. Information Fusion*, Chicago, IL, USA, 2011, pp. 1–8.

[47] X. Wang, Q. Z. Sheng, X. S. Fang, L. Yao, X. Xu, and X. Li, An integrated bayesian approach for effective multi-truth discovery, in *Proc. 24th Int. Conf. Information and Knowledge Management*, Melbourne, Australia, 2015, pp. 493–502.

[48] L. Blanco, V. Crescenzi, P. Merialdo, and P. Papotti, Probabilistic models to reconcile complex data from inaccurate data sources, in *Proc. 22nd Int. Conf. Advanced Information Systems Engineering*, Hammamet, Tunisia, 2010, pp. 83–97.

[49] G. Qi, C. C. Aggarwal, J. Han, and T. S. Huang, Mining collective intelligence in diverse groups, in *Proc. 22nd Int. World Wide Web Conf.*, Rio de Janeiro, Brazil, 2013, pp. 1041–1052.

[50] X. L. Dong, B. Saha, and D. Srivastava, Less is more: Selecting sources wisely for integration, in *Proc. 38th Int. Conf. Very Large Data Bases*, Istanbul, Turkey, 2012, pp. 37–48.

[51] T. Rekatsinas, X. L. Dong, and D. Srivastava, Characterizing and selecting fresh data sources, in *Proc. Int. Conf. Management of Data*, Snowbird, UT, USA, 2014, pp. 919–930.

[52] A. Silberschatz, H. F. Korth, S. Sudarshan, Functional-dependency theory, in *Database System Concepts Six Edition*, M. D. Bilecki, ed. New York, NY, USA: McGraw-Hill, 1997, pp. 338–348.

Jizhou Sun is a PhD student at School of Computer Science, Harbin Institute of Technology. He received the bachelor degree from Nanjing University of Aeronautics and Astronautics in 2009, and the master degree from China Aerospace Science and Industry Corporation in 2012. His research interests include data quality management and approximate computing on weak available massive data.

Jianzhong Li is a professor and PhD supervisor at School of Computer Science and Technology of Harbin Institute of Technology, China. He received the bachelor degree from Heilongjiang University in 1975. In the past, he worked as a visiting scholar at the University of California at Berkeley, as a staff scientist in the Information Research Group at the Lawrence Berkeley National Laboratory, and as a visiting professor at the University of Minnesota. His research interests include data management systems, sensor networks, and data intensive computing.

Hong Gao is a professor and PhD supervisor at School of Computer Science and Technology of Harbin Institute of Technology, China. She received the bachelor degree from Heilongjiang University in 1988, the master degree from Harbin Engineering University in 1991, and the PhD degree in computer science from Harbin Institute of Technology in 2004. Her research interests include wireless sensor networks, cyber-physical systems, massive data management, and data mining.

Hongzhi Wang is a professor and PhD supervisor at School of Computer Science and Technology of Harbin Institute of Technology, China. He received the bachelor, master, and PhD degrees in computer science from Harbin Institute of Technology in 2001, 2003, and 2008, respectively. His interested research area includes big data management, data quality, graph data management, and web data management.