Approximate Truth Discovery via Problem Scale Reduction

Xianzhi Wang[1], Quan Z. Sheng[1], Xiu Susie Fang[1], Xue Li[2], Xiaofei Xu[3], and Lina Yao[1]
[1]The University of Adelaide, Adelaide, SA 5005, Australia
[2]The University of Queensland, Brisbane, QLD 4072, Australia
[3]Harbin Institute of Technology, Harbin 150001, China
{xianzhi.wang, michael.sheng, xiu.fang, lina.yao}@adelaide.edu.au,
xiaofei@hit.edu.cn, xueli@itee.uq.edu.au

ABSTRACT

Many real-world applications rely on multiple data sources to provide information on their interested items. Due to the noises and uncertainty in data, given a specific item, the information from different sources may conflict. To make reliable decisions based on these data, it is important to identify the trustworthy information by resolving these conflicts, i.e., the truth discovery problem. Current solutions to this problem detect the veracity of each value jointly with the reliability of each source for every data item. In this way, the efficiency of truth discovery is strictly confined by the problem scale, which in turn limits truth discovery algorithms from being applicable on a large scale. To address this issue, we propose an approximate truth discovery approach, which divides sources and values into groups according to a user-specified approximation criterion. The groups are then used for efficient inter-value influence computation to improve the accuracy. Our approach is applicable to most existing truth discovery algorithms. Experiments on real-world datasets show that our approach improves the efficiency compared to existing algorithms while achieving similar or even better accuracy. The scalability is further demonstrated by experiments on large synthetic datasets.

Categories and Subject Descriptors

H.2.8 [**Information Systems**]: Database Management—*Data Mining*; I.2.m [**Computing Methodologies**]: Artificial Intelligence—*Miscellaneous*

Keywords

Truth discovery; problem scale reduction; recursive method; consistency assurance

1. INTRODUCTION

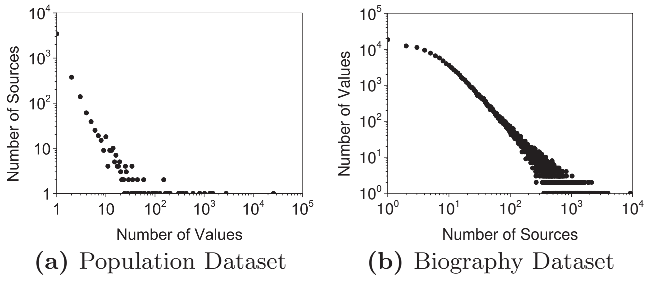
The increasing amount of data generated from the World Wide Web and the recently emerged Internet of Things (IoT) applications pose challenges to database research, characterized by the *volume*, *velocity*, *variety* and *veracity* features of Big Data. With advanced data extraction and collection technologies, information about a real-world entity can be obtained from various sources in both cyber and physical worlds. The data provided by distant sources are inherently unreliable, due to the uncertain data—noises, missing updates, missing values, transmission errors, or maliciously manipulated data. Thus, multi-source data for describing the same entity are liable to be conflicting. To support reliable decisions based on or to exploit the maximal value from multi-source data, it is critical to identify the trustworthy information from the multi-source inputs. This problem is non-trivial because the reliability of sources is often unknown *a priori* and ground truths are in most cases unavailable. To complicate the matter, the number of sources and entities in a truth discovery problem can be extremely large, which requires high scalability of truth discovery algorithms.

While the truth discovery problem has been studied from different perspectives [12], it remains inefficient. Waguih *et al.* [18] experimentally evaluated the performance of several truth discovery algorithms on three computing nodes on both real-world and synthetic datasets with various configurations, and concluded that most algorithms have efficiency problems. For example, both the algorithms based on Maximum Likelihood Estimation (MLE) and those on probabilistic graphical approaches were too computationally expensive to be applied to large scale problems. Moreover, to pursue higher accuracy, recent approaches incorporate more factors, such as the hardness of fact [7], the effect of random guess [16], and data sufficiency [10], making them increasingly complex and less efficient.

To the best of our knowledge, no studies have focused on improving the efficiency of truth discovery algorithms. We perceive that the issue in truth discovery is a hybrid technical problem, which inspires us to strategically reduce the problem scale and pursue approximate solutions in exchange for higher efficiency. Based on this insight, we propose an approximate approach focusing on reducing the problem scale. In a nutshell, we make the following contributions:

- We investigate the characteristics of real-world datasets and propose an approximate truth discovery approach, which groups sources and values to reduce the problem scale and considers the inter-group influence to improve truth discovery accuracy. The approach can achieve a personalized balance between the efficiency and accuracy of truth discovery algorithms according to a user-specified criterion.

CIKM'15, October 19–23, 2015, Melbourne, Australia.
© 2015 ACM. ISBN 978-1-4503-3794-6/15/10 ...$15.00.
DOI: http://dx.doi.org/10.1145/2806416.2806444.



(a) Population Dataset **(b)** Biography Dataset

Figure 1: The number of values claimed by each source: most sources claim very few values.

- We propose a number of content-based and mapping-based methods for grouping sources and values, respectively, to reduce the problem scale. In particular, we model content-based grouping as an optimization problem and solve it by a recursive algorithm with the *branch and bound* strategy.
- We present the methods for improving the truth discovery accuracy, by amending the estimated veracity scores of values without manual intervention.
- We conduct experiments using both real-world and synthetic datasets to evaluate the proposed approaches, which show the significant improvements of the efficiency of existing truth discovery algorithms without necessarily sacrificing the accuracy.

The rest of the paper is organized as follows. We discuss the observations that motivate our work and define the truth discovery problem in Section 2. Section 3 introduces our approach. The experimental results are reported in Section 4. We discuss the related work in Section 5 and give some concluding remarks in Section 6.

2. PRELIMINARIES

2.1 Observations and the Motivation

This work is motivated by two sources of observations: real world datasets and existing truth discovery algorithms.

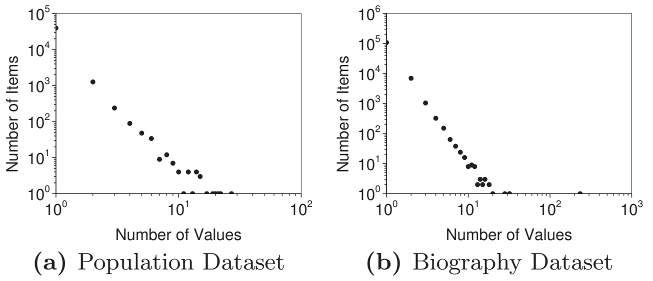
2.1.1 Real-World Datasets

We investigate the distribution of values on different sources and items in various real-world datasets (e.g., Figure 1 and Figure 2 show the results in *population* [14] and *biography* [14] datasets, respectively). We observe *long tail phenomenon* [10] in most investigated datasets, which has the following characteristics: i) most sources claim very few values and ii) most items have very few distinct values claimed.

Since the complexity of truth discovery is fundamentally determined by the problem scale—characterized by the number of links[1] between sources and values, an intuitive solution for improving truth discovery efficiency is to reduce the number of nodes (sources and values) and links. We focus on two types of nodes in developing our approach:

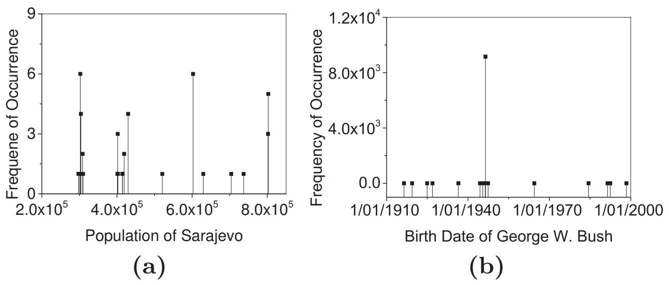
- *Sources that claim very few values.* While such sources make up a major portion of all sources, their association with values are sparse. This makes it more likely for different sources to claim the same set of values on the data items. Without extra information, truth discovery methods assess sources' reliability solely based on the sources' inputs. Hence, the sources that claim the same sets of values will always receive the same assessment results, and repetitively computing the reliability for each of those sources becomes unnecessary.
- *Items with large numbers of values.* Intuitively, when the values of an item become enormous, they easily become close. To verify this point, we investigate the value distribution of data items and observe varying distributions on different data items. For example, the claimed population sizes of *sarajevo* (Figure 3a) are distributed rather unevenly, with some values extremely close to each other and others well separated. In contrast, the claimed birth dates of *George W. Bush* (Figure 3b) are dominated by a single value. Indeed, in many practical cases, especially those where the optimal solution is difficult to reach, users do not usually care about the minor differences. Thus, we can provide approximate solutions and only offer more precise results per users' requests.

[1]A link exists between a source and a value only when the source claims the value.



(a) Population Dataset **(b)** Biography Dataset

Figure 2: The number of values claimed for each item: some data items have large numbers of values claimed.



(a) **(b)**

Figure 3: Examples of value distribution of the *population* and *biography* datasets: varying distributions for different data items (Note that some values are not shown because they fall out of the range of display).

2.1.2 Truth Discovery Algorithms

We investigate the truth probability[2] distribution of existing truth discovery algorithms during each cycle of iteration, if the algorithms require iteration, to examine if they

[2] *Truth probability* means the probability of being true, which is calculated by normalizing the evaluation results, or veracity score, of each distinct value.

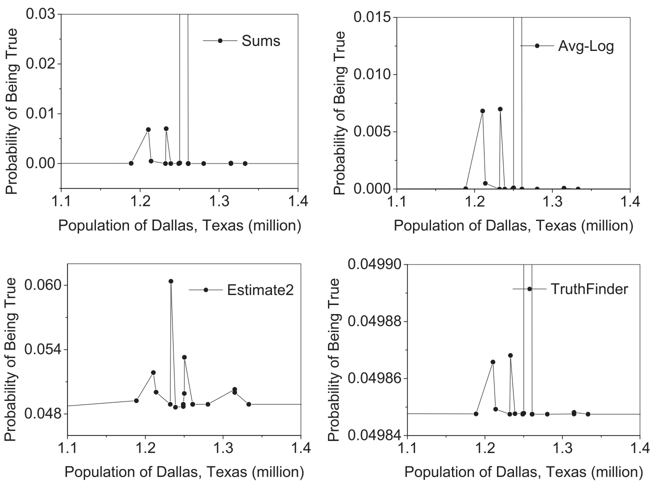


Figure 4: The distribution of truth probability achieved by four truth discovery algorithms during their third cycle of iteration over the different population sizes of Dallas, Texas.

produce consistent evaluation results. Here, by consistency, we mean *similar values should get similar truth probabilities*. For example, in determining a city's population size, two similar values, e.g., 8,390,482 and 8,416,535, should have similar truth probabilities. In contrast, the truth probability of 3,517,424 should be more different, since the value is less similar to the former two values. Enforcing consistent evaluation is crucial for ensuring the accuracy of truth discovery, as it improves the robustness of truth discovery algorithms in terms of smoothing the fluctuations and anomalies in the evaluation results. Intuitively, the truth discovery results cannot be justified without ensuring consistency because it would be insensible to assign very different truth probabilities to two values if they are extremely close.

Unfortunately, existing truth discovery algorithms lack consideration of such consistency, which obtain rather fluctuating truth probability distribution over the distinct values for the same item (as shown by the examples in Figure 4). Yin *et al.* [20] and Dong *et al.* [5] propose to improve the consistency by considering the inter-value influence in evaluating the values. They amend the evaluation results of an arbitrary value by:

$$\sigma^*(v) = \sigma(v) + \rho \cdot \sum_{v' \neq v} \sigma(v') \cdot sim(v', v) \qquad (1)$$

where v' and v are two different values for the same item, $sim(v', v) \in [0, 1]$ is their similarity; σ and σ^* are the veracity scores[3] of v before and after the amendment, respectively; $\rho \in (0, 1]$ is the influence factor which represents the influence strength.

Such amendment can significantly add to the computation load of truth discovery algorithms. It also requires manually defining the influence strength, which is sometimes tricky. Since in practice, we often regard a value as true as long as it is sufficiently close to the truth, a practical approximation may avail the amendment method in two aspects: i) reducing the computation load and ii) avoiding the necessity of manually determining the influence strength.

2.1.3 A Motivating Example

Based on above analysis and insights, we propose an approximate truth discovery approach, which groups sources and values to reduce the problem scale and considers the inter-group influence to improve the truth discovery accuracy. Now we illustrate our approach with an example.

EXAMPLE 1. *Suppose we want to corroborate the birth dates of some historical figures. Five data sources provide such information and only s_1 provided all the true values (Table 1). A naive voting would incur a computation load of $(6+5)+(6+4)*2 = 31$—for each of the three items (i.e., people), the algorithm performs two steps: i) scanning the six records to get the times of occurrence of each distinctive values, i.e., 5, 4, and 4 for the three items, respectively, and ii) checking through the five distinctive values to find out the most frequently occurring value as the estimated truth. The average accuracy is 0.5.*

By considering the inter-value influence, the computation load is increased to $(6 + \underline{P_2^5} + 5) + (6 + \underline{P_2^4} + 4) * 2 = 75$—between the two steps, the algorithm takes an additional step to calculate the mutual influence between each pair of different values, which is a permutation (denoted by P). The resulting accuracy is 1.

Now suppose a user only wants to know the birth years and months of these people instead of the exact dates. In this case, we can simply group the dates in the same year and month for each people, forming 3, 3, and 2 groups of dates for the three people, respectively. Moreover, we find s_2 and s_5 claim exactly the same groups of dates, so they can be grouped to form a joint source. Through such approximation, the computation load is decreased to $(5+3)*2+(5+2) = 23$ without veracity score amendment and $(5+\underline{P_2^3}+3)*2+(5+\underline{P_2^2}+2) = 37$ with amendment. The resulting accuracy is also 1. For now even if the user still wants an exact date, the approximation approach can still deliver the true values with an accuracy of 1—by selecting the most promising date from the most promising group for each people using the naive voting method.

Table 1: A motivating example: five sources provide information on the birth dates of three historical figures. Only s_1 provides all true values.

	e_1: G. Washington	e_2: A. Lincoln	e_3: N. Mandela
s_1	Feb. 22, 1732	Feb. 12, 1809	Jul. 18, 1918
s_2	Jul. 22, 1743	Feb. 20, 1809	Jul. 9, 1918
s_3	May 9, 1721	Jan. 20, 1813	Apr. 22, 1916
s_4	Feb. 27, 1732	Jan. 20, 1809	Apr. 16, 1916
s_5	Jul. 22, 1743	Feb. 12, 1809	Jul. 18, 1918
s_6	Feb. 15, 1732	Jan. 20, 1813	Jul. 18, 1918

2.2 Definitions and Notations

Let E be a set of entities. Each entity $e \in E$ is described by a set of attributes A.

Each pair (e, a), where $e \in E$, $a \in A$, represents a distinct data item.

Let S be a set of data sources. Each source $s \in S$ publish some values (that are potentially true) on a subset of attributes in A regarding a subset of the entities in E.

[3]Veracity score does not necessarily equal to the truth probability, but a higher score indicates a higher truth probability.

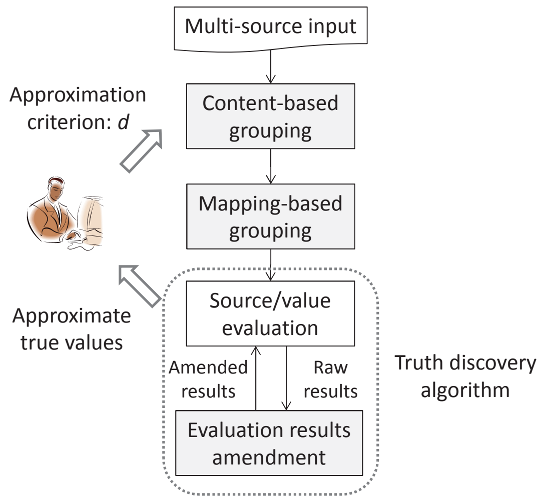


Figure 5: Approximate truth discovery

Let $V(e,a)$ be the set of all source-claimed values on data item (e,a). We consider the case where each item has only one true value, so at most one value in $V(e,a)$ can be true, while all the other values are false.

We denote by R a set of input records. Each record $r \in R$ describe a source's claim on a specific item (e,a), which is a 5-tuple: $< id, s, e, a, v >$, where id uniquely identifies the record and v is the value claimed by s on (e,a).

We assume all records from different sources have been transformed to a unified schema. Given a set of input records R published by a set of data sources S, the goal of truth discovery is to identify a single true value $v^* \in V(e,a)$, if the true value exists, for each attribute entity $e \in E$ and each attribute $a \in A$.

3. THE APPROACH

Our approach facilitates efficient truth discovery by reducing the problem scale and improves accuracy by enforcing consistent evaluation of the truth discovery results. It has three main components (as shaded in Figure 5) besides the truth discovery module: *content-based grouping*—for reducing the number of values, *mapping-based grouping*—for reducing the number of sources, and *veracity score amendment*—for enforcing consistency. The first two components are one-time jobs, while the last is incorporated into the truth discovery process.

- *Content-based grouping.* This component groups similar values according to a user-specified approximation criterion. Each resulting group will be regarded as a single value by truth discovery algorithms.
- *Mapping-based grouping.* This component groups the sources that claim the same portfolio of value-groups. Each resulting source-group will be regarded as a joint source by truth discovery algorithms.
- *Veracity score amendment.* This component amends the veracity scores of values by considering the inter-group influence of values. The amended results can be directly used for estimating the truth.

The following subsections will introduce each of the components, respectively.

3.1 Content-based Grouping

This component aims at reducing the problem scale by grouping similar values according to a user-specified approximation criterion. We assume user input as a personalized threshold (i.e., d in Figure 5) on the numerical distance between the true values estimated before and after using the content-based grouping methods, meaning any two values can be grouped together only when their distance is smaller than d. We notice that treating similar values as a group does not necessarily impair the truth discovery accuracy: i) the truth estimated based on single values may be inaccurate by itself and ii) due to the unreliable and unevenly distributed source inputs, the estimated true and false values could interleave in the spectrum of all values. Discovering truth based on the grouped values tends to be more robust in term of smoothing the local inconsistency and focusing on a global estimation. We also note that influence might still exists between the resulting value-groups. For example, Figure 6 shows an example of three values and the distances between them compared with the criterion d, where v_2 can be grouped with either v_1 or v_3. Whichever happens, the other value will still have an influence on v_2 from outside the group. In the worst case, if v_2 makes up a group of itself, it would be influenced by both the other values from outside of the group.

Referring to the philosophy of clustering, we expect each value to be maximally influenced by the (other) values of the same group while minimally influenced by the values of other groups, to preserve the accuracy of truth discovery. Therefore, we define the content-based grouping problem as follows: given a set of values V and the criterion d (which value is specific to each data item), find such a division of V, say $\boldsymbol{V_{opt}} = \{V_1, V_2, \ldots, V_n\}$, that:

$$Minimize \sum_{V_i, V_j \in \boldsymbol{V_{opt}}, i \neq j} inf(V_i \to V_j) \qquad (2)$$

$$subject\ to: \ \forall v_x, v_y \in V_i \ (\in \boldsymbol{V_{opt}}),\ |v_x - v_y| < d.$$

where V_i is a value-group, v_x and v_y are individual values, $inf(V_i \to V_j)$ quantifies the influence of V_i on V_j. The parameter d enables users to make personalized decisions on the degree of approximation. Given a larger d, less groups would be formed, and *vice versa*.

We prove the NP-Completeness of the content-based grouping problem as follows.

PROOF. The NP-Completeness proof proceeds via an efficient reduction from the Balanced Graph Partitioning problem (BGP) [1]. BGP aims at partitioning graphs into equal sized components while minimizing the number of edges between different components. The number of components immediately transforms into d when the values for each single data item are evenly distributed in their domain. If we replace the optimization objective of "the number of edges between different components" into "the mutual influence between different sub-graphs", BGP immediately transforms into the content-based grouping problem. □

While clustering represents a technique for set division, we avoid considering clustering as an appropriate solution for three reasons: i) most clustering methods cannot determine the optimal cluster number based on a single input such as d, ii) some do not guarantee to obtain the optimal results, and iii) it is hard to apply optimization techniques

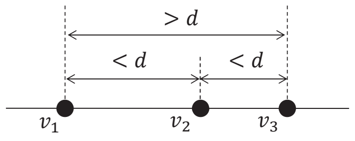


Figure 6: An example illustrating the circumstances that cause inter-group influence. Whichever another value (i.e., v_1 or v_3) v_2 is grouped with, or even v_2 makes up a group of itself, it will still be influenced by at least one other value from outside the group.

such as *branch and bound* to accelerate the clustering process. Another option would be based on graph models. We could take all distinctive values claimed for a specific data item as nodes to construct a complete graph. Then we can remove the edges longer than d (resulting in G) and partition G so that each resulting sub-graph is still a complete graph. In this way, the content-based grouping problem can be solved approximately by adopting existing solutions to the Maximum Clique Problem (MCP) [19]. Unfortunately, our problem differs from MCP in that, instead of pursuing the largest complete sub-graph, it pursues all complete sub-graphs and the minimal influence among those sub-graphs. Moreover, even if a MCP solution is applicable, we need to apply the MCP solution multiple times to find all the complete sub-graphs—by each time finding and removing the new maximum clique from the remaining graph. Besides severe computation load, this approach does not guarantee to produce the optimal grouping results.

Our problem is special in that all values are unidimensional and can thus be sorted for given specific data item. This enables us to employ novel heuristics to improve efficiency. We propose a recursive solution with *branch and bound* technique to optimally assign groups to the values. The components of this solution are described in Figure 7.

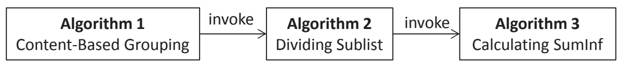


Figure 7: The invocation relation between the algorithms for content-based grouping.

Algorithm 1 is the entry of our content-based grouping solution. Given a specific data item, the algorithm first sorts the values for the item (line 1), and then compares the distance between each pair of adjacent values with the criterion d (lines 4-9). Whenever the distance between two values surpasses d, the list is divided in-between. Each sub-list is either regarded as a cluster of itself (if it contains a single value; lines 12-13) or applied the recursive method (otherwise; lines 14-18) to find the optimal group assignment. Since no influence exists between different sub-lists (due to the division criterion d), the grouping results for every sub-list together form the grouping results regarding the given data item.

Algorithm 2 recursively constructs the potentially optimal (and possibly partial) group division for the input sublist. It takes a preorder and depth-first traversal of the tree-like search space. Given a (partial) group division solution, which is a (either complete or incomplete) list of groups formed from the sublist, the algorithm first checks whether the inter-group influence of this (partial) solution exceeds the historical best—that is the bounding condition, which

Algorithm 1: Content-Based Grouping

```
   Input: a set of possible true values $V$; a tolerable deviation
          from the true value for the item $d$.
   Output: a set of value-groups $\{V_k | V = \cup V_k\}$.
1  $\{v_1, v_2, \dots, v_{|V|}\}$ ←sort the elements of $V$;
   /* Split the list of sorted values into sub-lists          */
2  $x \leftarrow 1$ ;                          // $x$ is the index for sub-list
3  $L_x \leftarrow \{v_1\}$ ;                      // initialize the first sub-list
4  foreach $i = 1, 2, \dots, |V| - 1$ do
5      if $|v_{i+1} - v_i| \leq d$ then
6          $L_x \leftarrow L_x \cup \{v_{i+1}\}$ ;    // add a value to the sub-list
7      else
8          $x$++;
9          $L_x \leftarrow \{v_{i+1}\}$ ;             // initialize a new sub-list
   /* Find the optimal group division for each sub-list      */
10 $LC \leftarrow \emptyset$ ;                             // the grouping results
11 foreach $L \in \{L_1, L_2, \dots, L_x\}$ do
12     if $|L| = 1$ then
13         $LC \leftarrow LC \cup \{L\}$;
14     else
           /* The initial (partial) group of interest         */
15         $C \leftarrow \{v | v$ is the first element of $L\}$;
           /* The initial (partial) solution of interest      */
16         $Solution_0 \leftarrow \{C\}$ ; // which is list of groups ordered
           by their time of addition to the list
17         $L \leftarrow L \backslash \{v\}$;
           /* initial as the maximal possible because no known
              solution currently exists                       */
18         $B_{best} \leftarrow Max\_Value$;
           /* call Algorithm 2                                */
19         $LC \leftarrow$
           $LC \cup$ DividingSublist($Solution_0$,$L$,$d(e,a)$,$B_{best}$);
```

is the minimal inter-group influence ever achieved by any complete group division solution. If it does, the solution will be abandoned; otherwise, the algorithm examines its completeness—whether the solution contains all the values of the sublist (line 2). The algorithm returns the solution and corresponding evaluation result when the solution is complete (line 4-5; $L = \emptyset$).

In case the solution is incomplete (line 6), the algorithm checks whether the first value of the sublist can be added to an existing group (line 10). If true, the process diverges into two branches: i) adding the value to the existing group (lines 11-14) and ii) setting up a new group for the value (lines 15-18). In the first case, Algorithm 2 invokes itself, taking the updated solution (with the new value added to the last group) and the updated sublist (with the first value removed) as inputs (line 12). If the solution returned by the self-invocation is not *null*—meaning the returned solution is better, the bounding condition (B_{best}) will be updated with the evaluation result of the returned solution. In the second case, Algorithm 2 invokes itself only under one condition: the distance between the new value (v) and the last element of the third last group in *Solution* is larger than d (line 17). In this case, Algorithm 2 takes the new solution (with a new group added) and the updated sublist (with the first value removed) as inputs (line 19). Differing from the first case, B_{best} will not be updated even if a better solution is found (the returned solution$\neq$ *null*)—information from the last branch can only be passed to its upper-layers (instead of its siblings) in the tree-like structure.

We prove that only under the condition specified in line 17 could the new solution possibly be optimal as follows.

PROOF. Given a (partial) solution that consists of four consecutive groups, C_i, C_{i+1}, C_{i+2}, and C_{i+3}, suppose the minimal distance between the elements of C_i and C_{i+3} is

Algorithm 2: Dividing Sublist

```
Input: the current (partial) solution of interest $Solution$, an
       ordered list of ungrouped values $L$; a tolerable deviation
       from the true value for the item $d$, the best evaluation
       result of all currently known solutions $B_{best}$.
Output: a couple $(Solution, B)$, where $Solution$ is the
        obtained grouping result and $B$ is the corresponding
        evaluation result.
1  $B_{current} \leftarrow$ CalculatingSumInf($Solution$)  //call Algorithm 3
2  if $B_{current} \geq B_{best}$ then
3      Return $null$ ;   // bound the branch with historical best
4  if $L = \emptyset$ then
5      Return $(Solution, B_{current})$ ;   // better solution found
6  else
7      $v \leftarrow$ the first element (i.e., a value) of $L$;
       /* get the current cluster in formation */
8      $Solution.last \leftarrow$ the last element (i.e., a set) of $Solution$;
9      $v_{lFirst} \leftarrow$ the first element (i.e., a value) of $Solution.last$;
       /* get alternative solutions */
10     if $|v - v_{lFirst}| \leq d$ then
           /* Choice 1: add as the new last element/value of
              the existing cluster */
11         $Solution.last \leftarrow Solution.last \cup \{v\}$;
12         $Result_1 \leftarrow$ DividingSublist($Solution$,$L\backslash\{v\}$,
           $d(e,a)$,$B_{best}$);
           // Suppose $Result_1 = (Solution_1, B_1)$
13         if $Result_1 \neq null$ then
14             $B_{best \leftarrow B_1}$ ;   // update known best result; new
               bounding condition is set
           /* Choice 2: initialize a new cluster by $v$ */
15         $Solution.thirdLast \leftarrow$ the third last element (i.e., a set)
           of $Solution$;
16         $v_{tlLast} \leftarrow$ the last element (i.e., a value) of
           $Solution.thirdLast$;
17         if $\nexists v_{tlLast} \vee |v - v_{tlLast}| > d$ then
18             $Solution \leftarrow Solution \cup \{v\}$ ;   // add as the new
               last element/cluster of $Solution$
19             $Result_2 \leftarrow$ DividingSublist($Solution$,$L\backslash\{v\}$,
               $d(e,a)$,$B_{best}$);
               // Suppose $Result_2 = (Solution_2, B_2)$
20             if $Result_2 \neq null$ then
21                 Return $Result_2$ ;   // better solution found
22     else
           /* only have one choice here */
23         $Solution.thirdLast \leftarrow$ the third last element (i.e., a set)
           of $Solution$;
24         $v_{tlLast} \leftarrow$ the last element (i.e., a value) of
           $Solution.thirdLast$;
25         if $\nexists v_{tlLast} \vee |v - v_{tlLast}| > d$ then
26             $Solution \leftarrow Solution \cup \{v\}$ ;   // add as the new
               last element/cluster of $Solution$
27             $Result_3 \leftarrow$ DividingSublist($Solution$,$L\backslash\{v\}$,
               $d(e,a)$,$B_{best}$);
               // Suppose $Result_3 = (Solution_3, B_3)$
28             if $Result_3 \neq null$ then
29                 Return $Result_3$;
30 Return $null$.
```

smaller than d. We can easily get a better solution by merging C_{i+1} and C_{i+2}. Thus the given (partial) solution could not be the optimal (partial) solution. □

For the case that the value cannot be added to an existing group (line 22), it is tackled in the similar way as we tackle the second branch above (lines 23-29). Note that, in Algorithm 2, we intentionally put the first case before the second, i.e., greedy selection —the first case does not add to the total inter-group influence, but the second case does.

Algorithm 3 evaluates the (partial) grouping solution for a specific sub-list. Given a group within the solution, the algorithm calculates the influence of at most five groups on this group—two precedent groups (line 6), two succeeding groups (line 8), if they exist, and the group itself (line 4). The algorithm returns the extent (i.e., a ratio) to which the group is influenced by other groups (line 9). The conditions in line 17 and line 25 of Algorithm 2 ensure a group is influenced by at most four other groups according to the user-specified criterion d.

Algorithm 3: Calculating SumInf

```
Input: A (partial) group assignment solution
       $Solution = \{C_i | i = 1, 2, \ldots, |Solution|\}$, which is a
       ordered list of value-groups.
Output: $SumInf$, the sum of influence between all pairwise
        groups.
1  while $C_i \in Solution$ do
2      $sum_{cur} \leftarrow 0$, $sum_{pre} \leftarrow 0$, $sum_{fol} \leftarrow 0$;
3      foreach $v \in C_i$ do
           /* influence by values of the same cluster */
4          $sum_{cur} \leftarrow sum_{cur} + \sum_{v' \in C_i \wedge v' \neq v} inf(v' \rightarrow v)$;
           /* influence by values of other clusters */
5          if $i > 1$ then
6              $sum_{pre} \leftarrow sum_{pre} + \sum_{v' \in C_{i-1}} inf(v' \rightarrow v)$;
7          if $i > 2$ then
8              $sum_{pre} \leftarrow sum_{pre} + \sum_{v' \in C_{i-2}} inf(v' \rightarrow v)$;
9          if $i < |LC|$ then
10             $sum_{fol} \leftarrow sum_{fol} + \sum_{v' \in C_{i+1}} inf(v' \rightarrow v)$;
11         if $i < |LC| - 1$ then
12             $sum_{fol} \leftarrow sum_{fol} + \sum_{v' \in C_{i+2}} inf(v' \rightarrow v)$;
13     $SumInf \leftarrow$
       $(sum_{pre} + sum_{fol})/(sum_{pre} + sum_{cur} + sum_{fol})$;
```

In Algorithm 3, the influence between two values, e.g., v' on v, is calculated as follows:

$$\begin{cases} \sigma = d \\ inf(v' \rightarrow v) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\left(v'-v\right)^2/2\sigma^2} \end{cases} \quad (3)$$

Different from traditional linear combination, we assume Gaussian density of the influences to a specific value. In this way, only the similar values would have a major influence on a value. In particular, for each item, Eq.(3) transforms d into an acceptable variable domain of Gaussian density function. We derive different influence domains to represent different levels of user confidence. For example, suppose a user proposes some d with a confidence degree of 0.8 ($\in [0,1]$), which approximately equals to the Area Under the Curve (AUC) of Gaussian density function within the domain of $(-\sigma, +\sigma)$, we get a σ which is equal to d in Eq.(3). Given a full confidence of approximately 1, the resulting influence domain would turn into $(-3\sigma, +3\sigma)$.

3.2 Mapping-based Grouping

This component aims at reducing the problem scale by grouping sources according to their associations with the value-groups. The mapping-based grouping (Algorithm 4) involves two steps: i) extracting (line 1-5) and sorting (line 6) each source's claims and sorting sources by their claim sizes (line 7), and ii) grouping the sources that make the same set of claims (lines 8-23): the algorithm only compares adjacent sources in the sorted list and makes further comparison only when the sources have the same claim size (Line 12).

Table 2 shows two examples of the grouping results on the *population dataset* [14], where four Wikipedia users provide the same population size for *New Orleans* and another four

Algorithm 4: Mapping-Based Grouping

```
Input: a set of records regarding value-groups $R' = \{r' =< id, s, e, a, C > | s \in S, e \in E, a \in A, C \in Solution(e, a)\}$.
Output: $\{S_k | S_k \subseteq S, \bigcup S_k = S \wedge \bigcap S_k = \emptyset\}$
1  foreach $s_i \in S$ do
2      $profile_i \leftarrow \emptyset$ ;                    // Initialize claims of $s_i$
3      foreach $r' \in R'$ do
4          if $r'.s$ equals $s_i$ then
5              $profile_i \leftarrow profile_i \cup \{<' r.e, r'.a, r'.C >\}$;
6      Sort triples of $profile_i$;
7  Sort $s_i$ by $\{|profile_i|\}$;
8  $i \leftarrow 1$; $k \leftarrow 1$ ;              // Initialize source & group index
9  while $i \leq |S|$ do
10     $S_k \leftarrow \{s_i\}$; $k{+}{+}$ ;                      // Initialize a group
11     $j \leftarrow i + 1$;
12     while $j \leq |S| \wedge |profile_i| == |profile_j|$ do
13         $isIdentical \leftarrow true$ ; // If two profiles are identical
14         foreach $m = 1, 2, \ldots, |profile_i|$ do
               /* $t_{i,m}$ is the $m$-th triple in $profile_i$          */
15             if $t_{i,m}.e\ != triple_{j,m}.e\ ||\ t_{i,m}.a\ != t_{j,m}.a\ ||$
               $t_{i,m}.C\ != t_{j,m}.C$ then
16                 $isIdentical \leftarrow false$;
17                 break;
18         if $isIdentical$ then
19             $S_k \leftarrow S_k \cup \{profile_j\}$;
20             $j{+}{+}$;
21         else
22             break;
23     $i \leftarrow j$;
```

users for *Mexico City*. The mapping-based grouping method reduces the total number of sources (or joint source) of the dataset from 4264 to 3874. It should be noted that, the grouped sources do not only receive the same assessment, but also have the same effect as the group of individual sources do before they are grouped together in the truth discovery process.

Table 2: Two examples of the mapping-based grouping results on the population dataset [14]: each of the two cities has four contributors claiming the same population size.

Source	Item	Value
2442135: Mikelj 4444: Infrogmation 395740: ArielGold 2558878: Duece22	New Orleans, Louisiana	484674
240649: Rune.welsh 3093230: Kwsn 0 (71.104.219.117) 111478: Mixcoatl	Mexico City	8720916

3.3 Veracity Score Amendment

This component aims at improving the accuracy of truth discovery by enforcing consistent evaluation of value-groups. The amendment (Algorithm 5) relies on pre-computed inter-group influence matrix to update the veracity scores, where $c(C_x)$ and $\hat{c}(C_x)$ are the original and amended veracity scores of value-group C_x, respectively, and $inf(C_x \to C_i)$ is the influence of C_x on another group C_i. The matrix is calculated in a similar way as what Algorithm 3 does in calculating the inter-group influence. The only difference is that instead of summing up the influences of all groups, here we separately calculate the influence of different groups and store this information in a pentadiagonal matrix. Algorithm 5 calculates the veracity score of a value-group by linearly combining the veracity scores of all relevant groups using their influences on this value-group as weights. We are aware that there exist more compact ways to store these quintuples, but they are out of the scope of this paper.

Algorithm 5 reduces the time complexity of inter-value influence computing by an order. Suppose $\forall e \in E$, $\forall a \in A$, $|V(e, a)| = V$ and $|LC(e, a)| = G$, where $LC(e, a)$ is the set of all value-groups for data item (e, a). The time complexity of traditional methods is $O(L \cdot V) \leq O(|E||A| \cdot V^2)$, where L is the number of links between sources and values. In contrast, Algorithm 5 achieves a time complexity of $O(L') \leq O(|E||A| \cdot G)$, where L' is the number of links between source-groups and value-groups. Furthermore, Algorithm 5 does not require configuring additional parameters, such as ρ in Eq.(1), but automatically incorporates the influence of different groups based on probabilities. The amended veracity scores remain in the same range of the original scores, therefore require no additional efforts for normalization.

Algorithm 5: Veracity Score Amendment

```
Input: a set of value-groups $LC(e, a)$, the correlation matrix for each item $M_{e,a}[|V|][|LC|]$, where $e \in E$, $a \in A$; a set of veracity scores $\{c(C) | C \in LC(e, a)\}$.
Output: a set of amended scores for each item $\{\hat{c}(C) | C \in LC(e, a)\}$.
1  foreach $e \in E$ and $a \in A$ do
2      foreach $C_i \in LC(e, a)$ do
3          $\hat{c}(C_i) \leftarrow \sum_{x \in \{i-1, i, i+1\}} inf(C_x \to C_i) c(C_x)$;
```

Our approach recommends the group with the highest veracity score as the estimated truth. But we can also estimate a single value as truth by using the weighted median method[4]. The procedure for weighting the values is similar to lines 2-12 in Algorithm 3: given each value v of the recommended group g, we calculate the influence of values on both inside and outside of the group, i.e., $inf^{(v)}_{inside}$ and $inf^{(v)}_{outside}$, respectively. Then, we assign v with weight $\frac{inf^{(v)}_{inside}}{inf^{(v)}_{inside} + inf^{(v)}_{outside}}$. After weighting and sorting the values, we can output the weighted median as the estimated truth.

4. EXPERIMENTS

In this section, we evaluate our approach using both real-world and synthetic datasets. We first discuss the experimental setup in Section 4.1 and then report the experimental results on real-world datasets in Section 4.2. Since the real-world datasets are limited in scale and diversity, we further evaluated the sensitivity and scalability of the proposed approach based on synthetic datasets.

4.1 Experimental Setup

We introduce the measures to evaluate the algorithms' performance and the baselines methods, respectively.

4.1.1 Performance Measures

We apply three measures to evaluate the performance. For all these measures, a smaller value indicates a better result.

[4]http://en.wikipedia.org/wiki/Weighted_median/

- *Mean Absolute Mean Error (MAE)*. The absolute linear distance between the estimated truth and the ground truth.

- *Root Mean Square Error (RMSE)*. Compared to MAE, RMSE amplifies and severely punishes large errors.

- *Computation Time*. The time required for the algorithm to produce a final result, which includes the time spent on possible pre-processing (such as source/value grouping), source/value evaluation, and possible post-processing (such as veracity score amendment).

4.1.2 Baselines

All the following algorithms are used in an unsupervised manner and compared on numerical datasets:

- *Voting*: for each item, outputs the value claimed by the most sources as the estimated truth.

- *Sums*: evaluates sources and values alternately from each other by iteration.

- *Avg-Log*: uses a non-linear function (a combination of logarithm and average functions) to assess sources.

- *Investment* [15]: evaluates values based on their contribution to the assessment of related sources.

- *2-Estimates* [7]: assumes that claiming one value indicates disclaiming (vote against) all unclaimed values.

- *TruthFinder* [20]: considers inter-value influence and evaluates each value by a probability.

- *AccuSim* [5]: evaluates a value by the *a posterior* probability and incorporating the inter-value influence.

- *Gaussian Truth Model (GTM)* [22]: applies generative models to estimate truths.

In this paper, we focus on discovering a single true value for each data item from the numerical data, but leave the incorporation of source dependency and other data types for future work. We implemented the algorithms in Java 7 and ran experiments on 3 PCs with Intel Core i7-5600 processor (3.20GH×8) and 16GB RAM.

4.2 Experiments on Real-World Datasets

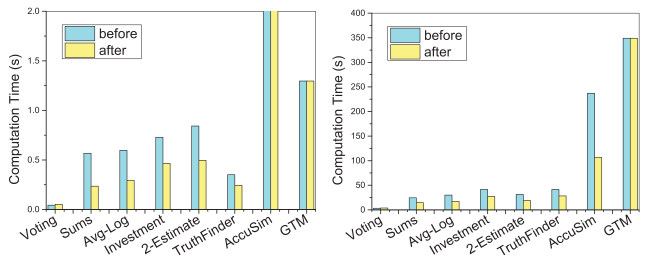
4.2.1 Datasets

We employ two real-world datasets[5] in the experiments and use the ground truth provided by the original contributors as gold standards.

The population dataset [14] contains 51,761 records. Each record describes a user's answer to a city's population size. After eliminating duplication and ambiguity, we got records about 4,264 sources claiming 49,956 values on 41,197 data items.

The biography dataset [14] contains 11,099,730 records. Each record describes a user's answer to the birth or death date of a person. After removing the irrelevant, duplicated, and ambiguous records, we merged the birth and death dates of the same person made by the same user and got 1,148,644 users and 3,211,983 claims related to 116,499 people.

[5]Until now, these are the only two public numerical datasets, which are also used in recent works [22, 14].



(a) *Population* dataset (b) *Biography* dataset

Figure 8: Comparison of computation time.

4.2.2 Comparison Results

Figue 8 shows the comparison of different algorithms in terms of their computation time before and after adopting our approach, based on the *population* and *biography datasets*. The results show that our approach reduces the computation time of the algorithms significantly, for some cases even by half. The effect is especially evident on the *population dataset*, where claims are sparse. Since our approach is not applicable to GTM, its computation time remains unchanged.

In Table 3 and Table 4, we summarize the performance of the algorithms before and after the adoption of our approach in terms of MAE and RMSE. The results show that our approach does not significantly decrease the truth discovery accuracy. Indeed, all algorithms achieve similar or even higher accuracy than their original implementations under the optimal configurations of d. This indicates that, if configured properly, our proposed approach does not only reduce the computation time, but also improves the truth discovery accuracy.

Table 3: Comparison on the *Population* dataset: the measures marked with asterisk represent the performance after adopting our approach; F# is the percentage of the estimated truths that have larger deviations from the actual truths than d.

Method	MAE	RMSE	MAE*	RMSE*	F#(%)
Voting	10327.2	126217.9	9823.4	124992.7	12.92
Sums	3026.25	17023.52	3008.47	17016.24	10.32
Avg-Log	2923.33	16923.36	2910.24	169431.3	9.21
Investment	1787.65	8797.43	1694.53	8634.38	13.84
2-Estimate	1652.29	8429.32	1609.98	**8339.99**	22.16
TruthFinder	1633.60	8824.09	1632.23	8823.09	**8.70**
AccuSim	1626.52	8718.10	**1498.59**	8596.26	9.26
GTM	**1523.29**	**8382.73**	1523.29	8382.73	8.87

Table 4: Comparison on the *biography* dataset: the meanings of the notations are the same as the ones in Table 3.

Method	MAE	RMSE	MAE*	RMSE*	F#(%)
Voting	237.35	4847.80	**213.53**	1772.63	22.65
Sums	396.69	5429.37	394.23	5427.46	18.23
Avg-Log	392.26	5296.62	382.33	5263.69	17.41
Investment	3858.90	26237.65	3294.73	24923.98	19.23
2-Estimate	234.42	**4723.53**	232.53	4694.97	20.83
TruthFinder	660.66	6959.72	661.39	6962.49	15.23
AccuSim	267.62	4844.46	253.28	**2823.53**	**11.46**
GTM	**228.19**	4831.53	228.19	4831.53	13.97

4.3 Experiments on Synthetic Datasets

4.3.1 *Datasets Preparation*

We generate synthetic data to evaluate the effectiveness of our approach on larger datasets. In particular, we generate synthetic records based on a real-world dataset, i.e., by adding new sources and data items to the *population dataset*.

We generate a new source by randomly selecting an existing source as reference—the new source will have exactly the same number of values on the same sets of items as the referred source. We then associate a source to the values in two steps: i) for each item, we divide the whole value range into a series of intervals, making sure each interval contains exactly one value. Given two adjacent intervals, their boundary is the average value of the two values that reside in the two intervals, respectively; ii) each time we add new association to a value: we first randomly select an interval which the value to be associated with are supposed to reside in, namely $\mathcal{I}$, and then associate the source with the existing value in $\mathcal{I}$ with a probability ρ, and with a random value within the interval $\mathcal{I}$ with a probability of $1-\rho$. We assume a slight increase in the total number of sources associated with the original values by defining $\rho = 0.01 (> 0)$. Data items are generated in a similar way.

4.3.2 *Scalability Experiments*

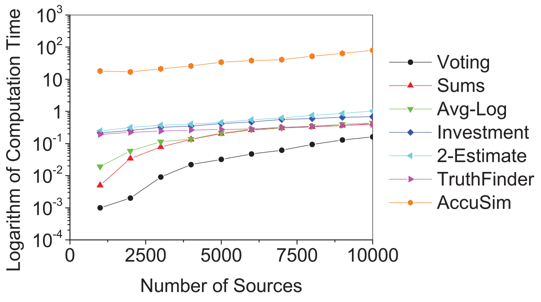
We conduct experiments to investigate the effect of our approach on improving the algorithms' scalability. We first fix the number of data items and vary the number of sources from 1,000 to 10,000, thus increasing the number of values for each data item. Then we fix the number of sources and vary the number of data items. Figure 9 shows how the reduced amount of computation time changes as the number of sources and the number of data items grow, after applying our approach. The results show the reduction amount keeps increasing as the two numbers grow. Since d remains fixed in both above experiments, the number of value-groups formed by our approach does not significantly change. But generally the more sources considered, the larger the source-groups tend to be, as the sources has a higher chance to be merged into an existing group. Experiments with respect to both above numbers reveal that the reduced computation time grows at lease linearly with the increased record size.

4.3.3 *Sensitivity Experiments*

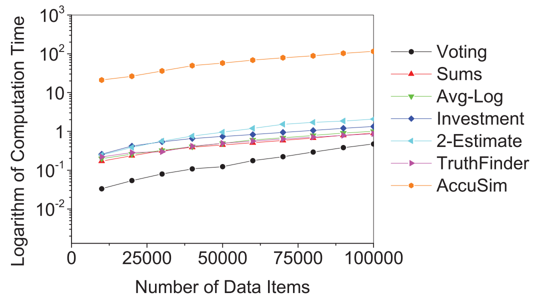
We study the impact of d to the effectiveness of our approach based on a synthetic dataset of 10,000 sources and 45,000 data items, where each item has on average 1 to 1,000 distinct values. Figure 10 shows the algorithms' performance with respect to a varying d within the range of $[0, SD/10]$, where SD is the standard deviation of all values for each data item—therefore, each item will have no more than 100 groups. Specially, when $d = 0$, the problem degrades to the circumstance of estimating the truths without adopting any grouping methods. We observe an exponential decrease in the computation time of all algorithms as d grows. As for the error rates, both MAE and RMSE tend to be low when d is small, but gradually increase as d grows.

5. RELATED WORK

Truth discovery has been a longstanding problem in the data quality research community [4, 13]. We classify existing



(a)



(b)

Figure 9: Computation time reduction under: (a) varying number of sources, (b) varying number of data items.

truth discovery approaches into four categories and review them in this section.

Primitive methods. These methods take the *mean*, *median*, or *majority voting* results of the values for each data time as estimated true values [3]. They are efficient but often inaccurate due to the neglect of varying reliabilities of sources in the real world.

Iterative algorithms. These methods jointly compute source reliability and confidence of value through an iterative procedure. Multiple ingredients, such as the inter-value influence [20], inter-value exclusion[7], source dependency [5], hardness of fact [7], as well as human knowledge [15] are incorporated to pursue more accurate estimation.

Probabilistic models. Many researchers model truths as latent variables in probabilistic graphical models to support truth discovery [16, 22]. Such models generally require tremendous computation time.

Optimization approaches. Yin and Tan [21] and Li *et al.* [11] respectively model the truth discovery problem as optimization problems and developed corresponding solutions.

Other related approaches include crowdsourcing [9] and web search [2, 17]. Crowdsourcing systems assess worker quality in accomplishing their crowdsourcing tasks while web search methods recommend trustworthy source links to answer user's queries. Truth discovery differs from crowdsourcing in pursuing only the factual truths and from web search in operating on well-formatted records instead of raw data.

The most related research is on source selection. Dong *et al.* [6] propose to select only partial sources for truth discovery, so as to reduce the integration cost while ensuring certain accuracy. Li *et al.* [9] present a framework for discovering the common characteristics of high quality workers, so as to facilitate the selection of high quality workers for future tasks. Gupta *et al.* [8] evaluate source quality and select only

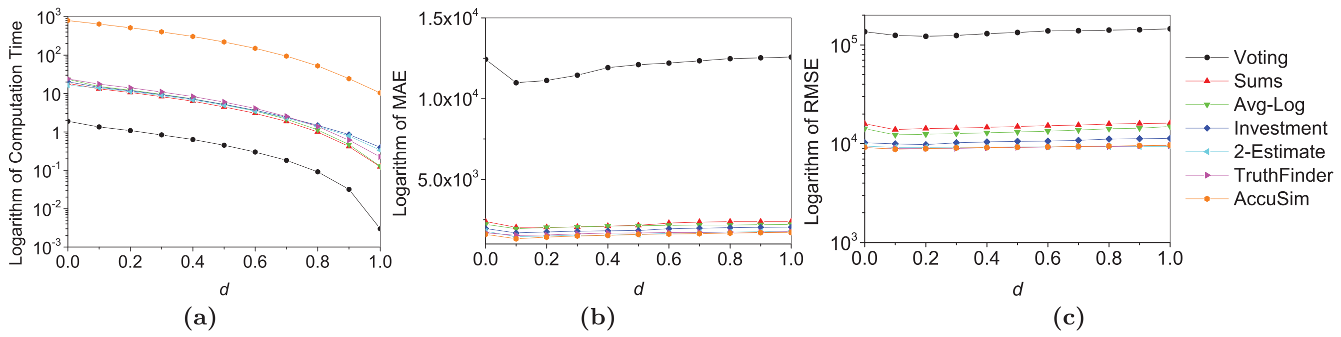


Figure 10: Comparison of algorithms' (a) computation time, (b) MAE, and (c) RMSE, under varying d.

trustworthy sources for each group of entities (obtained by clustering) to improve truth discovery accuracy. To the best of our knowledge, there is no previous work that specializes on improving the efficiency of truth discovery, nor utilizing grouping methods to reduce the problem size, which is the main contribution of this paper.

6. CONCLUSION

In this paper, we propose an approximate truth discovery approach, which improves the truth discovery efficiency via problem scale reduction. We develop content-based grouping and mapping-based grouping methods to reduce the sizes of sources and values, respectively. Veracity score amendment methods are used to improve the truth discovery accuracy with efficiency. The approach features approximate truth discovery according to a user-specified criterion, which can achieve a personalized balance between the efficiency and accuracy. The approach has been applied to several existing truth discovery algorithms. Experiments on both real-world and synthetic datasets demonstrate the effectiveness of our approach.

7. REFERENCES

[1] K. Andreev and H. Racke. Balanced graph partitioning. *Theory of Computing Systems*, 39(6):929–939, 2006.

[2] R. Balakrishnan and S. Kambhampati. Sourcerank: relevance and trust assessment for deep web sources based on inter-source agreement. In *WWW*, pages 227–236, 2011.

[3] J. Bleiholder and F. Naumann. Conflict handling strategies in an integrated information system. In *IJCAI Workshop on Information on the Web*, 2006.

[4] J. Bleiholder and F. Naumann. Data fusion. *CSUR*, 41(1):1–41, 2008.

[5] X. L. Dong, L. Berti-Equille, and D. Srivastava. Integrating conflicting data: the role of source dependence. *PVLDB*, 2(1):550–561, 2009.

[6] X. L. Dong, B. Saha, and D. Srivastava. Less is more: Selecting sources wisely for integration. *PVLDB*, 6(2):37–48, 2012.

[7] A. Galland, S. Abiteboul, A. Marian, and P. Senellart. Corroborating information from disagreeing views. In *WSDM*, pages 131–140, 2010.

[8] M. Gupta, Y. Sun, and J. Han. Trust analysis with clustering. In *WWW*, pages 53–54, 2011.

[9] H. Li, B. Zhao, and A. Fuxman. The wisdom of minority: discovering and targeting the right group of workers for crowdsourcing. In *WWW*, pages 165–176, 2014.

[10] 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. *PVLDB*, 8(4):425–436, 2014.

[11] 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 *SIGMOD*, pages 1187–1198, 2014.

[12] X. Li, X. L. Dong, K. Lyons, W. Meng, and D. Srivastava. Truth finding on the deep web: is the problem solved? *PVLDB*, 6(2):97–108, 2012.

[13] F. Naumann, A. Bilke, J. Bleiholder, and M. Weis. Data fusion in three steps: Resolving schema, tuple, and value inconsistencies. *IEEE Data Engineering Bulletin*, 29(2):21–31, 2006.

[14] J. Pasternack and D. Roth. Knowing what to believe (when you already know something). In *COLING*, pages 877–885, 2010.

[15] J. Pasternack and D. Roth. Making better informed trust decisions with generalized fact-finding. In *IJCAI*, pages 2324–2329, 2011.

[16] J. Pasternack and D. Roth. Latent credibility analysis. In *WWW*, pages 1009–1020, 2013.

[17] V. Vydiswaran, C. Zhai, and D. Roth. Content-driven trust propagation framework. In *SIGKDD*, pages 974–982, 2011.

[18] D. A. Waguih and L. Berti-Equille. Truth discovery algorithms: An experimental evaluation. *arXiv preprint arXiv:1409.6428*, 2014.

[19] Q. Wu and J.-K. Hao. A review on algorithms for maximum clique problems. *EJOR*, 242(3):693–709, 2015.

[20] X. Yin, J. Han, and P. S. Yu. Truth discovery with multiple conflicting information providers on the web. *TKDE*, 20(6):796–808, 2008.

[21] X. Yin and W. Tan. Semi-supervised truth discovery. In *WWW*, pages 217–226, 2011.

[22] B. Zhao and J. Han. A probabilistic model for estimating real-valued truth from conflicting sources. In *QDB*, 2012.