RCTD: Reputation-Constrained Truth Discovery in Sybil Attack Crowdsourcing Environment

Xing Jin
School of Cyberspace
Hangzhou Dianzi University
Hangzhou, Zhejiang, China
jinxing@hdu.edu.cn

Zhihai Gong
School of Cyberspace
Hangzhou Dianzi University
Hangzhou, Zhejiang, China
gongzhihai@hdu.edu.cn

Jiuchuan Jiang*
School of Information Engineering
Nanjing University of Finance and Economics
Nanjing, Jiangsu, China
jcjiang@nufe.edu.cn

Chao Wang
Research Center for Data Hub and Security
Zhejiang Lab
Hangzhou, Zhejiang, China
wangc@zhejianglab.com

Jian Zhang
School of Cyberspace
Hangzhou Dianzi University
Hangzhou, Zhejiang, China
zhangjian-hdu@hdu.edu.cn

Zhen Wang
School of Cyberspace
Experimental Center of Data Science and Intelligent Decision-Making
Hangzhou Dianzi University
Hangzhou, Zhejiang, China
wangzhen@hdu.edu.cn

Abstract

Sybil attacks are a prevalent concern within the realm of crowdsourcing, underscoring the significance of quality control in this domain. Truth discovery has been extensively studied to deduce the most trustworthy information from conflicting data based on the principle that reliable workers yield reliable answers. However, existing truth discovery approaches overlook the metric of workers' reputations, e.g., workers' historical approval rates on crowdsourcing platforms, despite being inflated and noisy, they offer a rough indication of workers' ability. In this paper, we first refine the approval rate using Wilson Lower Bound to enhance its confidence, and then mitigate its noise and inflation through a method based on ranking similarity. Specifically, we propose a method called *RCTD* (Reputation-Constrained Truth Discovery), which introduces a similarity metric between the rankings of workers' weights and the refined approval rates. This metric serves as a penalizing factor in the objective function of the truth discovery, restricting workers' weights to avoid excessively deviating from their historical reputation during the weight estimation process. We solve the objective function by introducing the block coordinate descent coupled with the heuristics approach method. Experimental results on real-world datasets demonstrate that our approach achieves more accurate inference of true results in the Sybil attack environment compared to the state-of-the-art methods.

*Corresponding Author

KDD '24, August 25–29, 2024, Barcelona, Spain.

ACM ISBN 979-8-4007-0490-1/24/08
https://doi.org/10.1145/3637528.3671803

CCS Concepts

• **Information systems** → **Trust; Reputation systems**.

Keywords

Crowdsourcing, Truth Discovery, Sybil Attack, Reputation

ACM Reference Format:
Xing Jin, Zhihai Gong, Jiuchuan Jiang, Chao Wang, Jian Zhang, and Zhen Wang. 2024. RCTD: Reputation-Constrained Truth Discovery in Sybil Attack Crowdsourcing Environment. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3637528.3671803

1 Introduction

Crowdsourcing involves using a distributed crowd of diverse knowledge backgrounds to perform human intelligent tasks that are challenging for machines, e.g., sentiment analysis [1] and information exploring [2]. In crowdsourcing, a task is often assigned to multiple workers. The inherent openness of crowdsourcing results in inconsistent skill levels among workers. Selfish individuals tend to spend less effort on quick rewards [3], while malicious workers seek to undermine the platform order. Both behaviors contribute to the unavoidable occurrence of Sybil attacks [4]. For instance, Google's crowdsourced navigation app, Waze, aggregates GPS values from users' devices to estimate real-time traffic. Malicious entities deploy lightweight scripts to emulate a substantial number of virtual vehicles referred to as Ghost Riders [5], which duly report fake data on the Waze server, leading the mapping app to mistakenly present incidents to users. Furthermore, on some crowdsourced data labeling platforms such as AMT [6] and Toloka [7], some approaches like anti-fraud systems and honeypots (golden task) are used to ensure real people are doing labeling tasks but not automation scripts. These automated scripts are typically orchestrated by selfish entities, employing tactics like sharing random results to surpass independent workers in voting or copying answers for quick

rewards [8]. These behaviors significantly undermine label aggregation in crowdsourcing platforms. Therefore, ensuring quality control in crowdsourcing becomes crucial, especially in the context of Sybil Attacks [9, 10].

One important method for addressing the crowdsourcing quality control problem is called Truth Discovery (TD). Dong et al. [11] proposed copy detection that attempts to identify different replication relationships between sources based on Bayesian analysis. The entity linking algorithm [12–14] assigns tasks based on the domains where workers showcase expertise. Probabilistic graphical methods [14–19] model workers and tasks, with each edge representing the relationship, and derive workers' model based on the principles of EM algorithm [20]. The *CRH* [21], proposed by Li et al. as a representative weighted-voting algorithm, is utilized for mining the truth from heterogeneous data, grounded in the principle that reliable sources provide reliable answers. In the *CRH*, workers receive a lower weight if their submitted answers deviate from the aggregated answers. In order to address the issue of strategic attackers controlling the aggregation results, Yuan et al. [11] proposed a cluster-based Sybil defense framework *SADU* [22], which clustered groups of workers based on the similarity of their behaviors. *SADU* identifies Sybil workers by assigning golden tasks to a group using a gold-inject [23] approach. A group is deemed Sybil if a significant number of its members provide inaccurate responses on golden tasks. Wang et al. proposed an online framework *TDSSA* [24] which runs extended truth discovery to update workers' weights and infer the truth of tasks in batch processing mode. *TDSSA* proposed a Sybil score that relies on the strategic Sybil behavior of workers and detects Sybil workers by allocating golden tasks based on Sybil scores, in contrast to the black-and-white detection of Sybil workers in *SADU*. Yuan et al. [25] studied the long-tail phenomenon in crowdsourcing data, and proposed a confidence-aware truth discovery *CATD*, which incorporates the number of tasks completed by the worker while estimating workers' weight. This capability enhances its accuracy in providing estimations for answers in datasets characterized by a long-tailed distribution.

Truth discovery methods effectively resist Sybil attacks, thereby enhancing mission quality. However, existing methods overlook workers' reputation metrics, such as their historical approval rates. Workers' approval rates provide a rough indication of workers' ability, although inflated and noisy [26], for instance, two workers have completed the same number of missions, one worker holds a 99.5% approval rate, while another has a 90% approval rate. To some extent, it is suggested that the former one has a higher work quality compared with the latter one. Therefore, deliberating on how to effectively leverage approval rates to enhance the results of truth discovery becomes crucial.

An intuitive idea to tackle the issue of inflated approval rates involves shifting our attention from the numerical values to the ranking. With this strategy, we can effectively alleviate the inflation of each worker's approval rate equally. An additional benefit of employing ranking lies in the ability to treat two or more workers as one element, which serves to diminish the influence of noise on workers' approval rates. In this study, *RCTD* (Reputation-Constrained Truth Discovery) is proposed by introducing a similarity metric as a penalty factor in the objective function of truth discovery. This penalty term is calculated by the similarity between the ranking of workers' weights and the ranking of approval rates. The penalty factor serves to prevent the estimated weight of workers from excessively deviating from their approval rates. In summary, our paper makes the following contributions:

(1) We propose an innovative truth discovery approach that leverages workers' historical approval rates to constrain the estimation of their abilities. Our method introduces a similarity metric between approval rates and workers' weights as a penalty term in the objective function of truth discovery. The primary goal is to ensure the estimated weights do not excessively deviate from the approval rates.
(2) Approval rates exhibit inflation, noise, and susceptibility to workers' historical task counts. In light of this, we first refine the approval rate by employing the lower bound of the Wilson Score Interval. Then, in order to mitigate noise, a concept of block similarity has been introduced, which gauges the correlation between weights ranking and approval rates ranking.
(3) We solve the objective function by iteratively estimating workers' weights and task answers using the block coordinate descent method. Specifically, in the computation of workers' weights, we present a heuristic algorithm based on a greedy approach.
(4) Experiments are conducted on four real datasets. Under various Sybil attack environments, the results indicate that *RCTD* outperforms the existing state-of-the-art algorithms. Furthermore, we showcase the effectiveness of our approval rate refined through ablation experiments.

The remainder of this paper is structured as follows: Section 2 outlines the problem and introduces a paradigm for solving it. In Section 3, we put forward a Reputation-Constrained objective function to define our problem. We present a greedy-based heuristic approach to solve it. Section 4 showcases the results of the experiment. We summarize the research in Section 5.

2 Preparation

2.1 Problem Formulation

In this paper, we consider a set $T = \{t_1, t_2, ..., t_i\}$ of single-choice crowdsourcing tasks. The single-choice task paradigm is extensively employed, such as audio classification [27] and entity recognition [28]. Numerous other task types can be transformed into a single-choice format like crowdsourcing navigation applications [29] and textual tasks [8]. Each task is structured with K candidate labels, and a constant number of N workers are employed to accomplish each task. We denote all the workers who complete the set of tasks T as $U = \{u_1, u_2, ..., u_j\}$. The submitted label of worker u_j on the task t_i is represented as $l_{i,j}$.

Sybil attack. In the online labeling markets, Sybil workers are typically dispersed among common workers [30, 31]. This implies that there could be Sybil accounts within the set U of workers. Sybil worker accounts are often orchestrated by automated scripts, enabling them to quickly gain profits [32, 33]. Strategic Sybil worker accounts are controlled by the same attacker and share random labels with a certain probability on a task [22, 24, 34]. This strategy is employed to deceive the system and evade detection, making the crowdsourcing system perceive the shared data from Sybil workers

as correct. Trusted workers may find themselves at a disadvantage due to the numerical superiority of the Sybil accounts. The presence of Sybil accounts poses challenges to crowdsourcing quality control. The setting of Sybil workers will be detailed in the experiment.

Reputation. In crowdsourcing platforms, each worker owns a historical approval rate of r_j that represents their reputations, e.g., in AMT [6], the approval rate is the proportion of his missions approved by requesters in the past. The approval rate value of workers is typically inflated. Requesters tend to give higher ratings even they are not so satisfied due to pressure from the platform [35] or fear of potential threats [26].

Problem. In the context of task set $T = \{t_1, t_2, ..., t_i\}$ with workers label $\mathcal{L} = \{l_{i,j}\}$ on each task t_i. Sybil workers illegally disguise themselves among common workers. Each worker account on the platform is associated with a historical approval rate r_j, serving as a measure of their reputation. We aim to leverage historical approval rates r_j to constrain the weight estimation in a Sybil attack scenario, thereby improving label aggregation accuracy.

2.2 Truth Discovery

Truth discovery is a category of methods used to address redundancy and conflicts in crowdsourced data. In truth discovery, each task is assigned to multiple workers, and trustworthy answers are inferred based on the workers' answers through techniques such as weighted voting.

Objective Function. Each worker s_j holds a weight w_j, with reliable workers are assigned with higher weights under the belief that they provide reliable answers. Based on the above intuition, the objective function of weighted voting [21] can be defined as:

$$\begin{aligned} &\underset{\mathcal{L}^*,\mathcal{W}}{argmin}\ f(\mathcal{L}^*,\mathcal{W}) = \sum_{j=1}^{J} w_j \sum_{i=1}^{I} d(l_{i,j}, l_i^*) \\ &s.t.\ \ w_j \sim \delta(\mathcal{W}) \end{aligned} \tag{1}$$

where $d(x, y)$ denotes the distance function, returning 0 if $x = y$ or 1 otherwise. By solving the objective function, it returns an aggregated label set $\mathcal{L}^* = \{l_i^*\}$ and workers' weight set $\mathcal{W} = \{w_j\}$. The item $\delta(\mathcal{W})$ reflects the distributions of the w_j. If weight w_j is unconstrained, we can easily let w_j be $-\infty$ to optimize the object function. The objective function can be interpreted as follows: when a worker's answer consistently deviates from the aggregated label, leading to a large sum of the value $d(l_{i,j}, l_i^*)$, the minimization of the objective function necessitates a smaller value for the worker's weight w_j. Hence, unreliable workers will be assigned less weight in subsequent tasks.

However, weighted voting is vulnerable to the manipulation of colluding Sybil workers. In a scenario where Sybil workers coordinately provide a consistent answer on a task, the labels of Sybil worker accounts outnumber the labels provided by independent workers, they could effectively override the contributions of independent workers and assert control over the aggregation result. In essence, without constraints on workers' weight, attacker entities gain the potential to manipulate the aggregation process. The subsequent sections will delve into our approach to address this challenge.

Table 1: Refined Approval with Wilson Score Interval

Worker	u_1	u_2	u_3
Approved Tasks/Total Tasks	9/10	90/100	180/200
Refined Approval Rate	0.76	0.84	0.86

3 METHODOLOGY

In this part, we initially refine workers' approval rate (Section 3.1) and formulate an innovative objective function with it (Section 3.2). The approval rate may be affected by a worker's historical ability which is with inflation and noise, so we propose a block similarity on weight ranking and approval rate ranking as a penalty term (Section 3.2.1 and 3.2.2). Lastly, to optimize this intricate objective function, we put forward a greedy-based heuristic method (Section 3.3).

3.1 Approval Rate Refinement

The number of assignments that a worker completes in his work history has a significant impact on the approval rate. For instance, considering two workers with the same approval rate of 100%, worker A completes 5 tasks with all 5 approved, and worker B completes 1000 tasks with all 1000 approved. The approval rate of worker B has a higher level of confidence in reflecting his ability compared to worker A.

Workers with few completed tasks should have lower confidence in their approval rate. To address this issue, an intuitive idea is to leverage the lower bound of Wilson Score Interval [36], a fitting metric for voting scenarios:

$$\mathcal{A}^* = \frac{\hat{ap} + \frac{z^2}{2H} - z\sqrt{\frac{\hat{ap}(1-\hat{ap})}{H} + \frac{z^2}{4n^2}}}{1 + \frac{z^2}{H}} \tag{2}$$

where $\mathcal{A}^*$ denotes the refined approval rate, $|H|$ is the number of tasks a worker completed in work history, $\hat{ap}$ denotes the approval rate, and z is the quantile of the normal distribution with the confidence level $1 - \alpha$. An example of refined approval rates is given:

Example 3.1. *Table 1 displays workers' refined approval rates. α set to 0.8. The numerical value of the approval rate remains the same at 0.9. The small-sized sample yields a smaller value after refinement.*

The advantage of using the lower bound of the Wilson Score Interval is that it takes workers' proficiency into account, thereby offering a more robust estimate. In cases with recently registered workers, the lower bound of the Wilson Score Interval helps to reduce the cold start problem to some extent, e.g., in AMT, new workers will be given an approval rate of 100%, and this method makes the approval rate estimation more fair to workers.

3.2 Reputaion-Constained TD

In Section 2.2, we have illustrated that the standard truth discovery can be vulnerable to attacks arising from collusion behaviors. A significant number of strategic Sybil workers may involved in completing a task. If there are no constraints on the workers' weights, it could cause the aggregated answers to deviate towards the Sybil workers. Therefore, our goal is to impose constraints on workers' weight. The approval rates can act as a gauge of their abilities. As

such, our goal is to formulate a penalty term $S(\mathcal{A}^*, \mathcal{W})$ to prevent the estimated weights from deviating excessively from approval rates. The objective function is defined as follows:

$$\underset{\mathcal{L}^*, \mathcal{W}^*}{argmin}\, f(\mathcal{L}^*, \mathcal{W}^*) = \sum_{j=1}^{J} w_j \sum_{i=1}^{I} d(l_{i,j}, l_i^*) - \lambda \cdot S(\mathcal{A}^*, \mathcal{W}) \quad (3)$$

where the penalty term $S(\mathcal{A}^*, \mathcal{W})$ is a function of refined approval rates set $\mathcal{A}^*$ and workers' weights set $\mathcal{W}$.

The rationale behind this lies in the fact that the approval rate, to some extent, reflects a platform's or a requester's endorsement of a worker's responses. However, the refined approval rate contains some level of noise and inflation, so it just provides a general indication of the potential performance of workers. In the subsequent section, we will introduce our approach for mitigating noise and inflation while formulating the penalty term.

3.2.1 Inflation Mitigation The numerical value of the approval rate is inevitably inflated in the online labor market. One possible reason is that raters may fear unknown retaliation or wish to avoid harming labor prospects [35]. Additionally, platforms often provide rewards or exert pressure on raters' evaluations [37], further contributing to the inflation of approval rates. As a result, the numeric value of approval rates may not precisely reflect the workers' abilities. When a requester selects laborers on platforms, aiming to mitigate the inflation issue, it is reasonable to compare the levels of approval rates among crowd workers.

Example 3.2. *Worker A achieved a 99% approval rate, while worker B attained 95%, and worker C, 90%. However, these approval rates do not directly reflect each worker's accuracy. Nevertheless, it's reasonable to infer that worker A exhibits superior labeling ability compared to worker B, and worker B outperforms worker C.*

Building on this concept, we consider the ordered ranking of the approval rates, denoted as $\mathcal{A}^*$. A penalty term $S(\mathcal{A}^*, \mathcal{W})$ is introduced to constrain weights' ranking $\mathcal{W}$ not deviate significantly from approval rates' ranking $\mathcal{A}^*$. In this paper, we employ *RBO* (Rank Based Overlap) [38] to measure the similarity between the two ordered sequences:

$$RBO(\mathcal{W}, \mathcal{A}^*, p) = k \sum_{d=1}^{J} p^d \frac{|I_{d,\mathcal{W},\mathcal{A}^*}|}{d} \quad (4)$$

where $\mathcal{W}$ and $\mathcal{A}^*$ be two infinite rankings. The number of intersection elements in the ranking $\mathcal{W}$ and $\mathcal{A}^*$ at deep d denote as $I_{|d,\mathcal{W},\mathcal{A}^*|}$. The decay parameter, represented by p, $0 < p < 1$, determines the element at depth d that we are interested in. When p is close to 0, more attention is given to the consistency of the elements near the top of the rankings, while p close to 1 means that equal consideration for all elements. Moreover, to ensure the penalty term ranges between 0 and 1, the coefficient k set to $\frac{1}{\sum_{d=1}^{J} p^d}$.

RBO used for similarity measurement exhibits several commendable properties. The similarity metric putting more attention on the forefront of workers in the approval rate ranking is prudent. Because in a standard crowdsourcing market, it is acceptable for Sybil workers not to have an exceptionally high approval rate. Another noteworthy advantage is the similarity value confined to the 0 to 1 range, making it well-suited for incorporation as a penalty term.

Table 2: The Rankings of Approval Rates and Weights

Rank Order	1	2	3	4	5	6
Rank by Approval rates	u_3	u_1	u_2	u_4	u_6	u_5
Rank by Weights	u_1	u_3	u_2	u_4	u_6	u_5

3.2.2 Weaken Noise The approval rates are subject to noise as they can be influenced by other factors, such as the subjective judgment of the requester [39, 40]. It is hard to ascertain the superior capabilities between two workers when their approval rates are closely aligned. Consider two workers who have approval rates of 97.5% and 97%, respectively. We cannot guarantee that the former has a higher quality compared with the latter. Hence, we treat adjacent workers as a unified entity and introduce the concept of *Block Similarity*, defined as follows:

$$S^*(\mathcal{A}^*, \mathcal{W}) = k^* \sum_{d=1}^{\lceil \frac{J}{n} \rceil} p^d \frac{|I_{nd,\mathcal{A}^*,\mathcal{W}}|}{nd}, (0 < p < 1, n = 1, 2, \ldots) \quad (5)$$

where n represents the step size. Compared to the original approach where each element is traversed individually, i.e., $n = 1$, the Block Similarity treats multiple workers as a single entity. This helps reduce the sensitivity of $S^*(\mathcal{A}^*, \mathcal{W})$ to the order of adjacent workers.

Example 3.3. *Table 2 gives the comparison of the similarity metric with step $n = 1$ and step $n = 2$. We calculate the rank similarity using Equation* (4)*, with p set to 0.9. It can obtain that $S(\mathcal{A}^*, \mathcal{W}) = k \cdot (\frac{0}{1} \cdot p^1 + \frac{2}{2} \cdot p^2 \ldots + \frac{6}{6} p^6) \approx 0.78$. When setting the step $n = 2$, and with Equation* (5)*, we get $S^*(\mathcal{A}^*, \mathcal{W}) = k^* \cdot (\frac{2}{2} \cdot p^0 + \frac{4}{4} \cdot p^1 + \frac{6}{6} p^2) = 1$.*

From the results, it can be observed that increasing the step size can reduce the sensitivity of similarity. The noise in reputation varies across different labor markets, where the choice of n needs to be determined by the dataset. Now, we can derive the objective function by substituting Equation (5) into Equation (3):

$$\underset{\mathcal{L}^*, \mathcal{W}}{argmin}\, f(\mathcal{L}^*, \mathcal{W}) = \sum_{j=1}^{J} w_j \sum_{i=1}^{I} d(l_{i,j}, l_i^*) - \lambda k^* \sum_{d=1}^{\lceil \frac{J}{n} \rceil} p^d \frac{|I_{nd,\mathcal{A}^*,\mathcal{W}}|}{nd} \quad (6)$$

3.3 Objective Function Solution

In solving the objective function, we utilize the block coordinate descent method. The iterative process involves updating the weights $\mathcal{W}$ and truth labels $\mathcal{L}^*$, gradually reducing the value of the objective function (6).

3.3.1 Truth Aggregation The label update process is straightforward since the similarity term is not associated with labels. As a result, the updating formula is the same as that used in *CRH* [21]:

$$l_i^*(t) \leftarrow \underset{l_i^*}{argmin} \sum_{j=1}^{J} w_j(t-1) \cdot d(l_{i,j}, l_i^*(t-1)) \quad (7)$$

it update the labels $l_i^*(t)$ with the results $w_j(t-1)$ and $l_i^*(t-1)$ obtained from the last iterative of the block coordinate descent.

3.3.2 Weight Updating During weight updating, the label l_i^* in Equation (6) is treated as a constant. Calculating the partial derivatives to worker weights is a challenge because the similarity component has a discrete nature. Traditional heuristic algorithms (such as *GA*) always take a considerable time to solve this problem and

are not conducive to convergence. Therefore, we propose a greedy-based approach to solve it:

Step 1 (Weight Swapping) . The objective function cannot be differentiated with respect to the weights, but we can incrementally adjust the weights in a greedy manner to minimize the objective function. We denote a descending ordered ranking of weights in iteration t as a vector $\mathcal{W}(t) = (w_1(t), w_2(t), ..., w_j(t))$. Note that, $w_j(t)$ represents the weight of the worker ranked at position j. Now, if we exchange the weights of the workers originally positioned at ranks k and $k-1$, we obtain a new vector denoted as $\mathcal{W}^-(t+1) = (w_1, ...w_k(t), w_{k-1}(t)..., w_j)$. Likewise, we can obtain vector $\mathcal{W}^+(t+1) = (w_1, ...w_{k+1}(t), w_k(t)..., w_j)$ as the replacement for the workers positioned at k and $k+1$. Substitute $\mathcal{W}(t)$ into objective function denoted as $f(\mathcal{W}(t))$, the same as $f(\mathcal{W}^-(t+1))$ and $f(\mathcal{W}^+(t+1))$. We propose the *Swapping Directional Derivative*:

Definition 3.1. *Swapping Directional Derivative refers to the increase in the objective function achieved by interchanging the weights of adjacent workers (the worker positioned forward or backward).*

$$\begin{aligned} \nabla^+_{w_k} &= f(\mathcal{W}^+(t)) - f(\mathcal{W}(t)), \\ \nabla^-_{w_k} &= f(\mathcal{W}(t)) - f(\mathcal{W}^-(t)). \end{aligned} \tag{8}$$

To guarantee the exchange of workers' weights achieves local optimality, we employ a greedy approach to swap adjacent workers' weights, as demonstrated below:

$$\begin{cases} swap(w_{k+1}, w_k), & \nabla^+_{w_k} < 0 \ and \ \nabla^-_{w_k} \le 0 \\ swap(w_k, w_{k-1}), & \nabla^+_{w_k} \ge 0 \ and \ \nabla^-_{w_k} > 0 \\ swap(w_k, \underset{w}{argmin}\{f(\mathcal{W}^-), f(\mathcal{W}^+)\}), & \nabla^+_{w_k} < 0 \ and \ \nabla^-_{w_k} > 0 \end{cases} \tag{9}$$

The replacement only happened between the weights but not the workers. Adjacent workers' weights are exchanged after replacement. Once the method for adjusting each worker's weight is established, we exchange adjacent workers' weight traverses from the one ranked at position 2 to $j-1$ according to Equation (9).

Step 2 (Sequential Constraint). The role played by the similarity penalty term $S^*(\mathcal{A}^*, \mathcal{W})$ is equivalent to imposing a sequential constraint on weight order. In other words, the objective function (6) is equivalent to the following form:

$$\begin{aligned} &\underset{\mathcal{W}}{\text{argmin}}\, f(\mathcal{W}) = \sum_{j=1}^{J} a_j w_j \\ &\text{s.t. } w_1 > w_2 > \ldots > w_{j-1} > w_j, \ \mathcal{W} \sim \delta(\mathcal{W}) \end{aligned} \tag{10}$$

where $a_j = \sum_{i=1}^{I} d(l_{i,j}, l_i^*)$ is a constant in the updating of workers' weight. Note that, a_j must follow a strict order corresponding to the worker ranked j in descending order.

To transform the constraints into parameters within the objective function, we set the difference of the weights between adjacent workers as $\beta_k > 0$, and represent the weights as the sum of β:

$$\begin{aligned} &w_j = \beta_j, \\ &w_{j-1} = w_j + \beta_{j-1} = \beta_j + \beta_{j-1}, \\ &w_{j-2} = w_{j-1} + \beta_{j-2} = \beta_j + \beta_{j-1} + \beta_{j-2}, \\ &\ldots \\ &w_1 = w_2 + ... + \beta_1 = \beta_j + \beta_{j-1} + ... + \beta_1, \\ &w_k = \sum_{i=k}^{j} \beta_i, \ (\beta_1, \beta_2, ..., \beta_j > 0) \end{aligned} \tag{11}$$

Then, the objective function (10) transforms into the following form:

Algorithm 1 Weight Updating

Input: The weight set of J workers in last iteration $\mathcal{W}$(t). The aggregated label set $\mathcal{L}$. Approval rate set $\mathcal{A}^*$.
Output: Updated worker weights: $\mathcal{W}$(t+1).

```
Rank worker by their weight in descending order 𝕎.
Calculating a_k = Σ_{i=1}^{I} d(l_{i,k}, l_i^*) for all workers.
for k ← 2 to J − 1 do
    Calculating ∇^+_{w_k} and ∇^-_{w_k} of worker k by Equation (8).
    Swapping the weight of worker k by Equation (9).
end for
for k ← 1 to J do
    Update the weight w_k(t+1) of worker k by Equation (15).
end for
return Worker weights: W(t+1).
```

$$\begin{aligned} \underset{\mathcal{B}}{argmin}\, f(\mathcal{B}) &= a_1 \sum_{k=1}^{j} \beta_k + a_2 \sum_{k=2}^{j} \beta_k + ... + a_j \beta_j \\ &= \beta_1 \cdot a_1 + \beta_2 \cdot (a_1 + a_2) + ... + \beta_j \cdot \sum_{k=1}^{j} a_k \end{aligned} \tag{12}$$

where $\mathcal{B}$ denotes the solution set of each β. To simplify Equation (12), let $b_k = \sum_{s=1}^{k} a_s$, which is a constant, where $k = 1, 2, ..., j$. In order to ensure the boundedness of the objective function, we constrain the maximum weight equal to 1, i.e., $\sum_{k=1}^{j} \beta_k = 1$. So we have:

$$\begin{aligned} &\underset{\mathcal{E}}{argmin}\, f(\mathcal{B}) = \sum_{k=1}^{j} b_k \beta_k \\ &s.t. \ \beta > 0, \ \sum_{k=1}^{j} \beta = 1 \end{aligned} \tag{13}$$

Equation (13) has a similar form as the objective function in *CRH*[21]. With the same method, we use Lagrange multipliers to solve this optimization problem:

$$\beta_k = -\log \frac{b_k}{\sum_{k=1}^{j} b_k} = -\log \frac{\sum_{s=1}^{k} a_s}{\sum_{k=1}^{j} \sum_{s=1}^{k} a_s} \tag{14}$$

From Equation (11) and (14), the weight of each worker can be obtained by:

$$w_k = \sum_{i=k}^{j} \beta_i = -\log \frac{\prod_{i=k}^{j} \sum_{s=1}^{i} a_s}{(\sum_{i=1}^{j} \sum_{s=1}^{i} a_s)^{j-k+1}} \tag{15}$$

Overall, the process of label aggregation includes two parts: Optimizing the weight sequence in a greedy manner and solving the objective function with the constraint of weight sequences. The pseudo-code of weights updating is summarized in Algorithm 1.

3.4 Algorithm Flow

As shown in Algorithm 2, our framework *RCTD* comprises the following parts: (1) Initializing the weights of all participating workers with $\frac{1}{J}$, where J is the number of workers and refining the approval rate of workers; (2) Employing the block coordinate descent method to iteratively update the weights of workers and the truth of labels. The algorithm stops when the truth in iteration $t+1$ is the same as those in the last iteration t or the iteration reaches a threshold. Besides, *RCTD* is no longer constrained by the golden task.

Algorithm 2 Reputation-Constrained Truth Discovery

Input: The answers of workers: $\{l_{i,j}\}$. The weight set $\mathcal{W}$. The aggregated label set $\mathcal{L}$. Approval rate set $\mathcal{A}^*$.
Output: Aggregated labels $\mathcal{L}$. The weight of workers $\mathcal{W}$.

```
Initialize the weight of workers as 1/J.
while Not all labels aggregated And Iteration < threshold do
    for The tasks completed in the current batch do
        l_i ← argmin_{l_i} Σ_{j=1}^{J} w_j · d(l_{i,j}, l_i^*).
    end for
    for All participants in the current batch do
        Update the w by Algorithm 1.
    end for
end while
return L and W.
```

4 PERFORMANCE EVALUATION

We evaluate the performance of *RCTD* algorithm across varying reputation scenarios and diverse Sybil attack properties. In Section 4.1, we describe the basic setup of our experiments. Section 4.2 compares algorithms in different Sybil attack properties. Section 4.3 evaluates the performance of *RCTD* in varying reputation scenarios. In Section 4.4, we conduct ablation experiments on two metrics. Experiments were implemented in JDK 1.8 and conducted on a computer with a CPU clocked at 2.0GHz with 16GB RAM. We place the source code of *RCTD* on GitHub[1].

4.1 Experiment Setting

4.1.1 Datasets To comprehensively evaluate the performance of *RCTD*, we select real-world crowdsourcing datasets with varying task counts, label sizes, etc. All tasks have ground truth available for evaluating the accuracy of algorithms.

(1) Sentiment Popularity (SP) [41] dataset records the answers of workers in a binary classification task using AMT platform. Workers were asked to label movie reviews from the internet, with positive or negative values.

(2) DOG [42] dataset focused on identifying the breed of a given dog. The dataset is extensively employed to evaluate the crowdsourcing quality control models in the presence of the Sybil attacker [22, 24].

(3) PosSent [9, 24] dataset is collected from AMT, with each task involving a tweet related to a company. Workers were asked to identify whether the tweet expressed a positive or negative sentiment toward to the corresponding company.

(4) Weather Sentiment (WS) [43] dataset categorizes tweets discussing weather sentiment into five levels: negative, neutral, positive, unrelated to weather, and indeterminate.

Detailed information about the data is presented in Appendix A.

4.1.2 Competing Methods To evaluate our algorithm's effectiveness in managing crowdsourcing quality in the presence of Sybil attacks, we select 5 competitive methods.

(1) MV [44, 45]. Majority voting treats all workers as equals, which is simple and fast for truth discovery. *MV* is regarded as a baseline method and helps to judge the rationality of the weight estimation in other weighted voting methods.

(2) CRH [21]. By utilizing the concept of reliable workers providing trustworthy facts, the algorithm formulates it as an optimization problem and iteratively resolves it. *CRH* algorithm demonstrates optimal performance primarily in scenarios with a limited number of Sybil workers.

(3) CATD [25]. Based on weighted voting, this algorithm leverages the count of missions completed by workers as an additional metric for assigning weights. The algorithm provides a solution for the long-tail distribution phenomenon observed in crowdsourcing data.

(4) SADU [22]. *SADU* classifies workers by establishing an average similarity threshold among them. Golden tasks are employed to identify the properties of a clustering.

(5) TDSSA [24]. The author introduces a batch-processing Sybil detection framework that assigns common tasks or golden tasks based on their collusion behavior. Golden tasks are allocated to workers for the verification of their identities.

(6) STDEL [46]. The author argues that the "Intelligent Sybils" may be intentionally completing only a small number of tasks to enhance the system's perceived weight of their contributions. The paper considers the quantity of completed tasks when estimating workers' weights.

Note that, *MV CRH*, *CATD*, and our method *RCTD* do not incorporate golden tasks, while *SADU*, *TDSSA* and *STDEL* utilize them.

4.1.3 Approval Rate Simulation In previous sections, we have discussed the inflation of workers' reputations, driven by factors like platform pressures, raters' bias, or their concern over potential consequences [35, 37]. This inflation is notably reflected in the discrepancy in the mean value between workers' approval rates and accuracy. Moreover, the distribution of workers' approval rates tends to be concentrated towards higher values, attributed to the aforementioned influences, as represented by a higher kurtosis. Detailed evidence supporting these assertions is provided in the Appendix B.3. Consequently, we propose using a tuple $\delta = (\Delta k, \Delta m)$ to represent the difference in approval rates and accuracy, where $\Delta k = k_{approval} - k_{accuracy}$ and $\Delta m = m_{approval} - m_{accuracy}$ denote the difference of kurtosis and mean, respectively. For a given tuple δ, we simulate approval rates through the following transformation equation:

$$a_j = \min\{\ \log(acc_j + C) + \mathcal{N}(\mu, \sigma), 1\ \} \tag{16}$$

where a_j and acc_j denote the approval rate and accuracy of worker j, respectively. If worker j is a common worker, the acc_j represents the accuracy of himself, and if worker j is a Sybil worker, we set acc_j as the mean accuracy across all workers. Our goal is to maintain a negligible distinction in the approval rate between independent workers and Sybil workers. The logarithmic transformation $log(acc_j + C)$ is employed to increase kurtosis, and C is a constant utilized for adjustment. Gaussian noise $\mathcal{N}(\mu, \sigma)$ serves to simulate both the real-world noise and inflation in approval rates. The approval rates truncate exceeding 1. We take $\delta = (\Delta k, \Delta m)$ as an indicator for simulated approval rates.

4.1.4 Attacker Injection To facilitate a comparative analysis of algorithm performance across diverse Sybil attack scenarios, most

[1] https://github.com/east207/RCTD

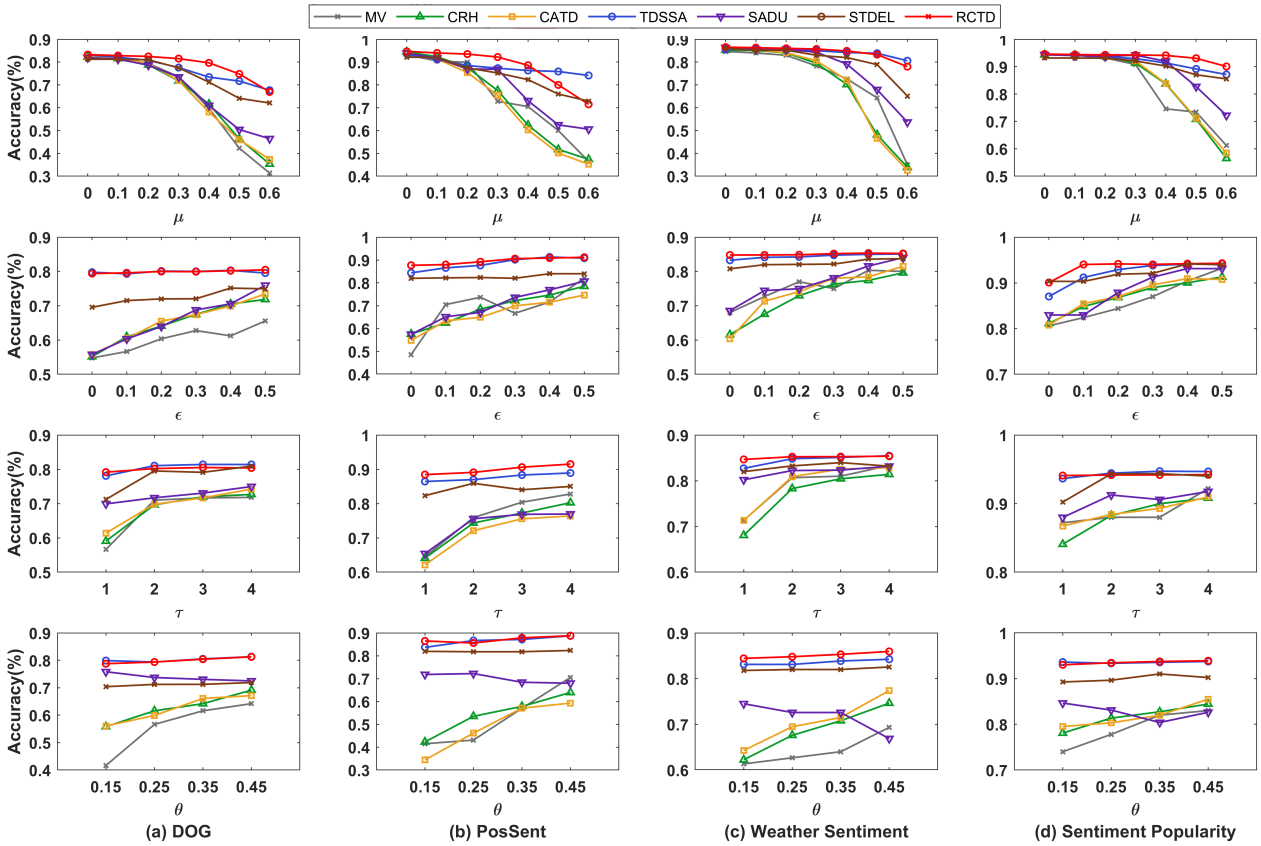

Figure 1: The average accuracy of label aggregation influenced by Attacker Properties with the *RCTD*.

existing research on crowdsourcing quality control sets Sybil attacks by replicating the answers of an attacker entity [8, 22, 24, 47]. However, real-world Sybil attacks are more intricate, for instance, Sybil accounts do not consistently submit the same answers, and the quantity of Sybil accounts controlled by an attacker entity remains uncertain. Therefore, we inject the Sybil accounts by manipulating several parameters as the method mentioned in *SADU* [22], *TDSSA* [24] and *STDEL* [46]. We utilize the following indicators to offer a more comprehensive portrayal of Sybil workers.

(1) The proportion μ of Sybil workers among all the workers.

(2) Sybil worker accounts collude with a probability of $1-\epsilon$, conversely, ϵ represents the probability that Sybil accounts submit answers independently.

(3) τ attacker entities. In crowdsourcing labor markets, it is not limited to one single attacker entity.

(4) Average accuracy of Sybil worker accounts θ. Sybil workers do not always share random answers which makes them easily detectable.

The setup of Sybil workers and the regularization parameter in experiments will be presented in Appendix B.2 and B.1 respectively.

4.2 Experiment on Varying Attacker Properties

In this part, we evaluate the aggregation accuracy of our method across four Sybil proportions. We established the approval rates using the default parameter configuration of $\delta = (\Delta k = 1.5, \Delta m = 0.15)$. Comparable outcomes were observed across alternative experimental setups, however, due to space constraints, they are omitted herein. Experimental results are depicted in **Figure** 1.

We observe that *RCTD* has the highest aggregation accuracy across the most tested Sybil properties. Our approach also outperforms the state-of-the-art algorithm *TDSSA*, achieving slightly higher accuracy but without requiring golden tasks. *CRH* and *CATD* exhibit similar performance, as *CATD* accounts for the long-tail distribution of task counts completed by workers and builds upon the *CRH* concept. However, the long-tail phenomenon in selected crowdsourcing datasets is not prominently featured. That is the reason *STDEL* performs slightly less favorably compared to *TDSSA*. Furthermore, *SADU*, which utilizes golden tasks for qualification verification, demonstrates a noticeable improvement in accuracy compared to methods lacking qualification verification. *SADU* classifies workers into clusters rather than considering the potential of each worker's individual identity, resulting in lower accuracy compared to *SADU*.

The effect of μ. As the proportion of Sybil worker accounts μ increases, all algorithms consistently exhibit a decline in accuracy, and the accuracy reduction for *RCTD* is relatively slight. In the **PosSent** dataset, when the Sybil proportion is high above 0.4, *RCTD*'s accuracy declines. The reason is that the average accuracy of independent workers in the **PosSent** dataset is a less-than-ideal

67%, which is only slightly higher than the random selection strategy of 50% for binary tasks. This implies that the responses from independent workers lack concentration, thereby allowing Sybil workers to prevail in voting when their quantities are substantial.

The effect of ϵ. As collusion diminishes, algorithms consistently demonstrate varying accuracy improvement. Notably, in scenarios of intense collusion, such as $\epsilon = 0$, *RCTD* outperforms existing algorithms in accuracy.

The effect of τ. A higher value of τ indicates the presence of more Sybil attacker entities in a single request. We assume that these entities are mutually independent and that only multiple Sybil worker accounts controlled by the same attacker entity share the data. Consequently, the transition from $\tau = 1$ to $\tau = 2$ results in a halving of the collusion level, which enables ordinary workers to regain control over the labels. As τ continues to increase, the improvement in accuracy is not significant.

The effect of θ. Under the condition of other Sybil properties remaining unchanged, a large value of θ implies that Sybil worker accounts share more accurate answers, making it hard to distinguish between common workers and Sybil workers. Experimental results show that an increment in θ leads to an unpredictable slight decline in the accuracy of *SADU*. This is attributed to greater difficulty in classifying workers within *SADU* as their behaviors become more similar. We set the maximum value of θ to 0.45, which is still below the average accuracy of independent workers. This suggests that Sybil workers continue to have a negative impact on the aggregated results. Our algorithm demonstrates slightly higher accuracy compared to *TDSSA* across different datasets. The average Accuracy of the Four Datasets is shown in Appendix C.

4.3 Evaluation on Varying Reputation Scenario

We assess the performance of *RCTD* across different scenarios. Utilizing a tuple $\delta = (\Delta k, \Delta m)$ to represent diverse scenarios, we iterate through μ and C to find a desired transformation based on Equation (16). Experimental results are illustrated in **Figure** 2.

Across the four datasets, we observe a trend where the accuracy of *RCTD* initially increases and then declines as the kurtosis of the approval rate rises, which appears to contradict conventional understanding. This phenomenon can be attributed to the differing distributions of approval rates between independent workers and Sybil accounts. When Δk is relatively low, the discrimination between these distributions is minimal. However, as Δk increases, the discrimination becomes evident. Thus, it appears that reputation inflation does not invariably harm the market.

As the inflation level Δm increases, there is a consistent decline in label aggregation accuracy across the four datasets. **Figure** 2 indicates that the impact on the aggregation accuracy of *RCTD* is minimal when Δm remains low. This is because workers' approval rates experience comparable inflation on the same platform, and *RCTD* effectively mitigates the impact of inflation for each worker by employing a ranking metric of approval rates. However, as the approval rate gradually inflates towards 100%, we can observe a discernible drop in label aggregation accuracy on some datasets. This occurs because the approval rates among workers converge towards a similar value, making it difficult to distinguish between workers with varying levels of approval rates.

4.4 Ablation Experiment

4.4.1 More Fair Evaluation We incorporate the historical approval rate in our method *RCTD*, a feature absent in the benchmarks, which may introduce an apparent bias against the baseline algorithm. To address this issue, we initialize workers' weights for *CRH*, *CATD*, *TDSSA* and *STDEL* algorithms using historical approval rates, represented by $w_j = a_j / \Sigma_{j=1}^{|U|} a_j$. It is important to clarify that *MV* assumes the weights of all workers are equal which could not be initialized with any value, and *SADU* utilizes a clustering-based method for workers' behavior that is not involvement of weights, so precluding the use of approval rates. Therefore, we compare our algorithm *RCTD* with baseline algorithms (*CRH*, *CATD*, *TDSSA* and *STDEL*) that use the approval rate for initialization. Note that, the reputation used here for initialization is unprocessed. **Figure** 3 demonstrates that a straightforward method utilizing approval rates only marginally improves accuracy (The gray bar in the figure as shown, e.g. CRH-R).

4.4.2 Effect on Approval Rate Refinement We conduct ablation experiments on the method of refined approval rates by the lower

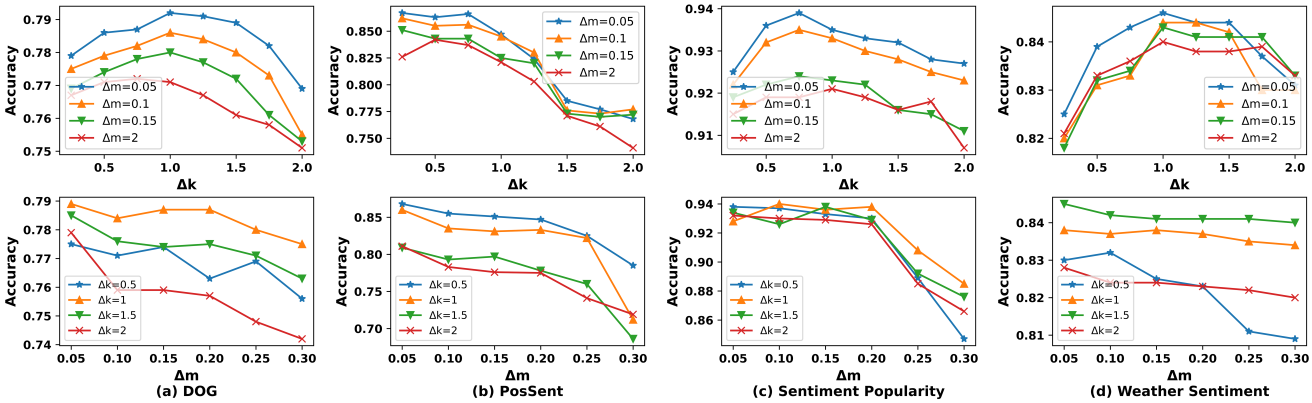

Figure 2: Evaluation on Varying Reputation Scenario by different values of Kurtosis and Mean. Attacker Propoties set with default value as $\mu = 0.4, \tau = 1, \epsilon = 0.1, \theta = 0.25/0.45$**.**

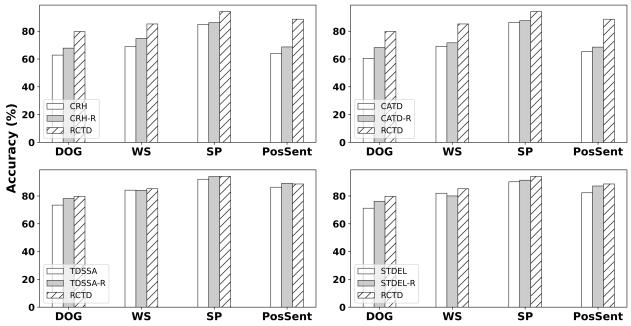

Figure 3: A Fairer Comparison: RCTD vs Benchmarks Initialized with Workers' Approval Rates.

bound of the Wilson Confidence Score interval. For each of the four datasets, our algorithm introduced approval rates before and after revision, with the latter set at a confidence level of $\alpha = 0.5$. We set the approval rates using $\delta = (\Delta k = 1.5, \Delta m = 0.15)$. The experimental results shown in **Table** 3, following 100 parallel experiments, reveal that the group of refined approval rates consistently achieves higher accuracy, and more pronounced enhancement is observed in the **DOG** dataset. This can be attributed to the fact that a considerable number of workers in the **DOG** dataset answer only a limited number of questions (an average of 74 tasks answered per worker out of 109 workers, but 31 workers completing fewer than 10 tasks). Workers completing a small number of tasks have less confidence in their accuracy, underscoring the efficacy of employing the refined approval rates for improved accuracy.

Table 3: Ablation Experiment on Refined Approval Rate

Settings	Accuracy of *RCTD* (100 Rounds, No Sybil)			
	DOG	PosSent	WS	SP
Refined	**83.60%**	**94.05%**	**87.33%**	**94.80%**
No Refined	**82.88%**	**93.88%**	**87.00%**	**94.80%**

5 CONCLUSION

In this paper, we explored the issue of Sybil attacks in the crowdsourcing environment, resulting in the unreasonable estimation of workers' weights in many algorithms. We proposed a method called *RCTD* that utilizes workers' historical reputation data (approval rates) as constraints for estimating workers' weights. We studied the deficiencies of the approval rate, such as inflation, noise, and the confidence level of the worker's approval rate. Initially, we refined the approval rates and computed the similarity between weight ranking and approval rate ranking in blocks. Then, we defined the similarity metric as a penalty term in the objective function. We solved the objective function with a combination of heuristic and block coordinate descent methods. Experimental results demonstrated that *RCTD* achieved higher accuracy in varying reputation scenarios. Additionally, *RCTD* eliminated the need for golden task detection to assess the quality of workers to some extent. In other domains, such as crowdsensing, *RCTD* can be expanded to build a reputation system and suggest a method for managing reputation.

6 ACKNOWLEDGMENTS

This research is in part supported by the Key R&D Programs of Zhejiang (2022C01018), Zhejiang Provincial Natural Science Foundation of China (LTGG23F030004), National Natural Science Foundation of China (62176080, 62002092, 42201510, and U21B2001), and National Social Science Funds of China (22BGL261).

References

[1] Bing Liu. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167, 2012.
[2] Ju Fan, Meiyu Lu, Beng Chin Ooi, Wang-Chiew Tan, and Meihui Zhang. A hybrid machine-crowdsourcing system for matching web tables. In *2014 IEEE 30th International Conference on Data Engineering*, pages 976–987. IEEE, 2014.
[3] Luke Gottlieb, Gerald Friedland, Jaeyoung Choi, Pascal Kelm, and Thomas Sikora. Creating experts from the crowd: Techniques for finding workers for difficult tasks. *IEEE Transactions on Multimedia*, 16(7):2075–2079, 2014.
[4] John R Douceur. The sybil attack. In *International workshop on peer-to-peer systems*, pages 251–260. Springer, 2002.
[5] Gang Wang, Bolun Wang, Tianyi Wang, Ana Nika, Haitao Zheng, and Ben Y Zhao. Ghost riders: Sybil attacks on crowdsourced mobile mapping services. *IEEE/ACM transactions on networking*, 26(3):1123–1136, 2018.
[6] Amazon mechanical turk. https://www.mturk.com. Accessed: February 9, 2024.
[7] Toloka crowdsourcing datasets. https://toloka.ai/datasets/?datasets-category=crowdsourcing#datasets. Accessed: February 9, 2024.
[8] Lingyun Jiang, Xiaofu Niu, Jia Xu, Dejun Yang, and Lijie Xu. Incentive mechanism design for truth discovery in crowdsourcing with copiers. *IEEE Transactions on Services Computing*, 15(5):2838–2853, 2021.
[9] Yudian Zheng, Guoliang Li, Yuanbing Li, Caihua Shan, and Reynold Cheng. Truth inference in crowdsourcing: Is the problem solved? *Proceedings of the VLDB Endowment*, 10(5):541–552, 2017.
[10] Florian Daniel, Pavel Kucherbaev, Cinzia Cappiello, Boualem Benatallah, and Mohammad Allahbakhsh. Quality control in crowdsourcing: A survey of quality attributes, assessment techniques, and assurance actions. *ACM Computing Surveys (CSUR)*, 51(1):1–40, 2018.
[11] Xin Luna Dong, Laure Berti-Equille, and Divesh Srivastava. Integrating conflicting data: the role of source dependence. *Proceedings of the VLDB Endowment*, 2(1):550–561, 2009.
[12] Yudian Zheng, Guoliang Li, and Reynold Cheng. Docs: a domain-aware crowdsourcing system using knowledge bases. *Proceedings of the VLDB Endowment*, 10(4):361–372, 2016.
[13] Fenglong Ma, Yaliang Li, Qi Li, Minghui Qiu, Jing Gao, Shi Zhi, Lu Su, Bo Zhao, Heng Ji, and Jiawei Han. Faitcrowd: Fine grained truth discovery for crowdsourced data aggregation. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 745–754, 2015.
[14] Gianluca Demartini, Djellel Eddine Difallah, and Philippe Cudré-Mauroux. Zencrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In *Proceedings of the 21st international conference on World Wide Web*, pages 469–478, 2012.
[15] David Karger, Sewoong Oh, and Devavrat Shah. Iterative learning for reliable crowdsourcing systems. *Advances in neural information processing systems*, 24, 2011.
[16] Qiang Liu, Jian Peng, and Alexander T Ihler. Variational inference for crowdsourcing. *Advances in neural information processing systems*, 25, 2012.
[17] Viet-An Nguyen, Peibei Shi, Jagdish Ramakrishnan, Narjes Torabi, Nimar S Arora, Udi Weinsberg, and Michael Tingley. Crowdsourcing with contextual uncertainty. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3645–3655, 2022.
[18] Zhou Zhao, Da Yan, Wilfred Ng, and Shi Gao. A transfer learning based framework of crowd-selection on twitter. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1514–1517, 2013.
[19] Hyun-Chul Kim and Zoubin Ghahramani. Bayesian classifier combination. In *Artificial Intelligence and Statistics*, pages 619–627. PMLR, 2012.
[20] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society: series B (methodological)*, 39(1):1–22, 1977.
[21] Qi Li, Yaliang Li, Jing Gao, Bo Zhao, Wei Fan, and Jiawei Han. Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 1187–1198, 2014.
[22] Dong Yuan, Guoliang Li, Qi Li, and Yudian Zheng. Sybil defense in crowdsourcing platforms. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1529–1538, 2017.

[23] Xuan Liu, Meiyu Lu, Beng Chin Ooi, Yanyan Shen, Sai Wu, and Meihui Zhang. Cdas: a crowdsourcing data analytics system. *arXiv preprint arXiv:1207.0143*, 2012.

[24] Yue Wang, Ke Wang, and Chunyan Miao. Truth discovery against strategic sybil attack in crowdsourcing. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 95–104, 2020.

[25] Qi Li, Yaliang Li, Jing Gao, Lu Su, Bo Zhao, Murat Demirbas, Wei Fan, and Jiawei Han. A confidence-aware approach for truth discovery on long-tail data. *Proceedings of the VLDB Endowment*, 8(4):425–436, 2014.

[26] Marios Kokkodis. Reputation deflation through dynamic expertise assessment in online labor markets. In *The World Wide Web Conference*, pages 896–905, 2019.

[27] Ana Elisa Méndez Méndez, Mark Cartwright, Juan Pablo Bello, and Oded Nov. Eliciting confidence for improving crowdsourced audio annotations. *Proceedings of the ACM on Human-Computer Interaction*, 6(CSCW1):1–25, 2022.

[28] Tim Finin, Will Murnane, Anand Karandikar, Nicholas Keller, Justin Martineau, Mark Dredze, et al. Annotating named entities in twitter data with crowdsourcing. In *Proceedings of the NAACL Workshop on Creating Speech and Text Language Data With Amazon's Mechanical Turk*, 2010.

[29] Muhammad Ajmal Azad, Samiran Bag, Simon Parkinson, and Feng Hao. Trustvote: Privacy-preserving node ranking in vehicular networks. *IEEE Internet of Things Journal*, 6(4):5878–5891, 2018.

[30] Marti Motoyama, Damon McCoy, Kirill Levchenko, Stefan Savage, and Geoffrey M Voelker. Dirty jobs: The role of freelance labor in web service abuse. In *20th USENIX Security Symposium (USENIX Security 11)*, 2011.

[31] Gang Wang, Christo Wilson, Xiaohan Zhao, Yibo Zhu, Manish Mohanlal, Haitao Zheng, and Ben Y Zhao. Serf and turf: crowdturfing for fun and profit. In *Proceedings of the 21st international conference on World Wide Web*, pages 679–688, 2012.

[32] Carsten Eickhoff and Arjen P de Vries. Increasing cheat robustness of crowdsourcing tasks. *Information retrieval*, 16:121–137, 2013.

[33] Ryan Kennedy, Scott Clifford, Tyler Burleigh, Philip D Waggoner, Ryan Jewell, and Nicholas JG Winter. The shape of and solutions to the mturk quality crisis. *Political Science Research and Methods*, 8(4):614–629, 2020.

[34] Zhao Ben Yanbin, Zhi Yang, Chriso WIlson, and Xiao Wang. Uncovering social network sybils in the wild. In *The 11th ACM SIGCOMM*. ACM SIGCOMM, 2011.

[35] Apostolos Filippas, John Joseph Horton, and Joseph Golden. Reputation inflation. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, pages 483–484, 2018.

[36] Edwin B Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927.

[37] Nan Hu, Jie Zhang, and Paul A Pavlou. Overcoming the j-shaped distribution of product reviews. *Communications of the ACM*, 52(10):144–147, 2009.

[38] William Webber, Alistair Moffat, and Justin Zobel. A similarity measure for indefinite rankings. *ACM Transactions on Information Systems (TOIS)*, 28(4):1–38, 2010.

[39] Panagiotis G Ipeirotis. Analyzing the amazon mechanical turk marketplace. *XRDS: Crossroads, The ACM magazine for students*, 17(2):16–21, 2010.

[40] John Joseph Horton and Lydia B Chilton. The labor economics of paid crowdsourcing. In *Proceedings of the 11th ACM conference on Electronic commerce*, pages 209–218, 2010.

[41] Sentiment popularity - amazon mechanical turk dataset. https://eprints.soton.ac.uk/376544/. Accessed: February 9, 2024.

[42] Dengyong Zhou, Sumit Basu, Yi Mao, and John Platt. Learning from the wisdom of crowds by minimax entropy. *Advances in neural information processing systems*, 25, 2012.

[43] Weather sentiment - amazon mechanical turk dataset. https://eprints.soton.ac.uk/376543/. Accessed: February 9, 2024.

[44] Michael J Franklin, Donald Kossmann, Tim Kraska, Sukriti Ramesh, and Reynold Xin. Crowddb: answering queries with crowdsourcing. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 61–72, 2011.

[45] Aditya Ganesh Parameswaran, Hyunjung Park, Hector Garcia-Molina, Neoklis Polyzotis, and Jennifer Widom. Deco: declarative crowdsourcing. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1203–1212, 2012.

[46] Dejia Lin, Yongdong Wu, and Wensheng Gan. Sybil-resistant truth discovery in crowdsourcing by exploiting the long-tail effect. In *2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 1545–1550. IEEE, 2022.

[47] Zhou Su, Qifan Qi, Qichao Xu, Song Guo, and Xiaowei Wang. Incentive scheme for cyber physical social systems based on user behaviors. *IEEE Transactions on Emerging Topics in Computing*, 8(1):92–103, 2017.

A Dataset Summary

Table 4: The properties of four datasets

Datasets	PosSent	WS	DOG	SP
Tasks with Ground Truth	1000	300	807	500
Worker Count	85	110	109	143
Total Labels	20,000	6000	8070	10,000
Workers Number per Task	20	20	10	20
Label Size	2	5	4	2
Average Accuracy	0.687	0.703	0.696	0.894

B Detailed Parameter Settings

B.1 Regularization Term

Parameters need to be determined in the regularization term including the coefficient of the term λ, the decay factor $0 \leq p \leq 1$, and the step of n. When the decay factor p is set to 0, only the top-ranked worker (determined by step, not only one worker) is considered and the similarity score is either 0 or 1, as p approaches 1, the evaluation of each worker is nearly equal. In the objective function of *RCTD*, if more Sybil workers exist on the platform, more attention should be paid to those workers with a high approval rate, so a small number of decay factor p could be selected. When a larger step n is chosen, the model becomes less sensitive to noise, and when n is set to 1, the order of each adjacent worker's approval rates affects the objective function. A large step of n is suggested to be selected if the reputation is inflated severely, or if many factors unrelated to workers' skills impact the approval rate. On a crowdsourcing platform, the selection of p and n is influenced by the environment. The parameter λ is used to adjust the relationship between the weighted voting part of the algorithm and the regularization term, we suggest selecting the average number of tasks completed by each worker, e.g. select $\lambda = 10000/143 = 70$ for **SP** dataset, the purpose of our doing this is to ensure that the scope of weighted voting items and punishment term remains consistent. We use grid-search to select parameters p and n that maximize the accuracy of truth inference, where we control the attributes of the Sybil attacker to be $\mu = 0.4, \epsilon = 0.1, \tau = 1, \theta = 0.25/0.45$. We repeat experiments 50 times for each pair of parameters, the average accuracy record in **Figure** 4.

B.2 Sybil Account Settings

In the experiment, we varied one of the parameters while maintaining the rest of the parameters unchanged. We set default Sybil properties as $\mu = 0.4, \epsilon = 0.1, \tau = 1, \theta = 0.25/0.45$. We set $\mu = 0.4$ because the accuracy of the truth inference decreases fast for most algorithms while the proportion of Sybil workers varied from 0.3 to 0.4, and $\tau = 1$ and $\epsilon = 0.1$ are also set to enhance the degree of collusion, the same parameter setting was employed in the study [24]. Sybil workers do not always share the same and random labels for quick rewards, so we varied the accuracy of Sybil workers from 0.15 to 0.45 with step 0.1 for convenience. $\theta = 0.25$ is used to set the dataset of **WS** and **DOG** because they are 5 choose 1 or 4 choose 1 task, the number of 0.25 is near to their random answer results. We do not set the $\theta = 0.2$ for the **WS** because it leads to difficult control variables. The same goes for the other two datasets. The parameter setting of the Sybil attackers is shown in **Table** 5.

Table 5: Paramter Settings of Sybil Attackers

Sybil Properties	μ	τ	ϵ	θ
PosSent, SP	**0.4**	**1**	**0.1**	**0.45**
WS, DOG	**0.4**	**1**	**0.1**	**0.25**
Range	**0: 0.1: 0.6**	**1: 1: 4**	**0: 0.1: 0.5**	**0.15: 0.1: 0.45**

B.3 Approval Rate Settings

One of the criteria for assessing workers' historical approval rates is based on whether their answers align with the voting results. On tasks that are prone to errors, true answers are often elusive due to errors made collectively by workers. Consequently, on tasks with unknown ground truth, workers' estimated accuracy tends to be higher than their actual accuracy. **Figure** 5 illustrates workers' estimated accuracy on unknown truth tasks (by majority voting) as well as their accuracy on ground truth tasks in the **TlkAgg5** dataset [7]. The distribution of workers' accuracy on unknown truth tasks results demonstrated a larger kurtosis and slightly higher accuracy than ground truth tasks.

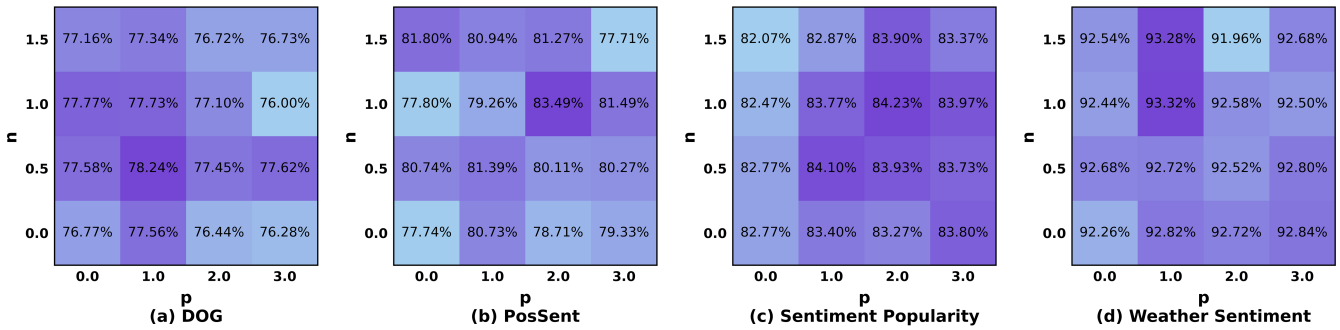

Figure 4: The performance of *the RCTD* method under different pair of parameter settings.

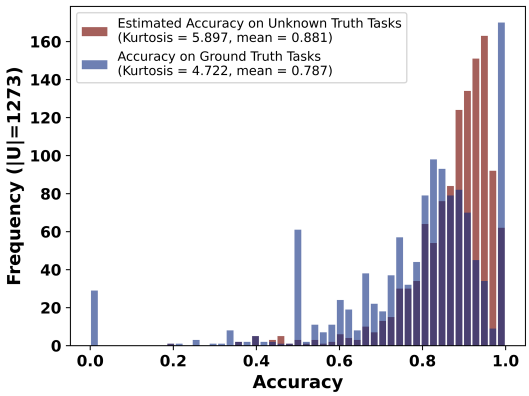

Figure 5: Distribution of Workers' Accuracy on Ground Truth Tasks vs Estimated Accuracy on Unknown Truth Tasks.

C Average Accuracy

Table 6: The Average Accuracy of the Four Datasets, Changes One Property while Keeping the Others Unchanged.

Attacker Properties		Average Accuracy(%) of the Four Datasets				
		CRH	CATD	TDSSA	STDEL	RCTD
μ	0	0.891	0.890	0.892	0.882	0.898 ↑
	0.1	0.883	0.881	0.883	0.879	0.894 ↑
	0.2	0.861	0.857	0.871	0.867	0.890 ↑
	0.3	0.800	0.800	0.857	0.844	0.884 ↑
	0.4	0.695	0.681	0.838	0.814	0.868 ↑
	0.5	0.543	0.535	0.826	0.765	0.828 ↑
	0.6	0.433	0.434	0.798 ↑	0.714	0.766
ϵ	0	0.639	0.629	0.836	0.807	0.885 ↑
	0.1	0.689	0.701	0.853	0.815	0.866 ↑
	0.2	0.731	0.729	0.862	0.821	0.871 ↑
	0.3	0.762	0.762	0.872	0.821	0.874 ↑
	0.4	0.781	0.777	0.879 ↑	0.842	0.877
	0.5	0.803	0.801	0.874	0.841	0.878 ↑
τ	1	0.688	0.704	0.852	0.807	0.866 ↑
	2	0.776	0.778	0.868	0.858	0.872 ↑
	3	0.799	0.798	0.874	0.854	0.876 ↑
	4	0.813	0.811	0.876	0.858	0.879 ↑
θ	0.15	0.596	0.586	0.851	0.808	0.857 ↑
	0.25	0.660	0.640	0.856	0.812	0.858 ↑
	0.35	0.689	0.692	0.863	0.815	0.868 ↑
	0.45	0.730	0.723	0.870	0.817	0.875 ↑

D Time Complexity

The time complexity of *RCTD* is proportional to the input size of the problem $|\mathcal{L}|$, where $|\mathcal{L}|$ is the total labels in a dataset, e.g. 6000 in **WS** dataset (see **Table** 4). The method of *RCTD* consists of b batches, while each batch refers to a specific count of tasks being completed, and for each batch, the time consumption depends on r iterations of label updating and weight updating. The weight updating iterates over the batch workers so the time complexity can be represented as $O(|U_b|)$, where the $|U_b|$ is the number of workers involved in a batch. So, the time consumption in a batch can be represented as $O(|U_b| \cdot |\mathcal{L}_b|)$, where $\mathcal{L}_b$ is the input label count in a batch. Overall, the time complexity of *RCTD* is bounded by $O(r \cdot |U| \cdot |\mathcal{L}|)$, and the time complexity could be $O(|U| \cdot |\mathcal{L}|)$ if we select a constant value of iteration r that *RCTD* converges in less or equal than r iterations.

The solution of algorithms *CRH*, *CATD*, *TDSSA* and *STDEL* all apply the idea of block coordinate descent methods to iteratively update the weights and labels. The computational complexity of these methods is linear with respect to the input size of the question, i.e. $O(\mathcal{L})$. Although their computational complexity is theoretically consistent, the *CATD* and *STDEL* cost much time because they take into account not only the label but also the number of tasks completed by workers while updating the weights, the weight updating process cost more time to converge towards the local optimum of the objective function. In the experiment, we keep $\mu = 0.4$, $\tau = 1$, $\epsilon = 0.1$ and $\theta = 0.25/0.45$, the approval rates set with $\delta = (\Delta k = 1.5, \Delta m = 0.15)$, the average running time is shown in **Table** 7. *RCTD* has an acceptable running time with higher accuracy than other algorithms.

Table 7: The Running Time of Experimental Results with Comparison Methods

Dataset	Running Time of the Algorithms (ms)						
	MV	CRH	CATD	TDSSA	SADU	STDEL	RCTD
WS	<1	45	245	47	2502	339	69
DOG	<1	134	656	153	4648	827	206
SP	1	113	694	142	5267	905	205
PosSent	1	280	690	193	6446	876	177