Truth Inference in Crowdsourcing: Is the Problem Solved?

Yudian Zheng [†], Guoliang Li [#], Yuanbing Li [#], Caihua Shan [†], Reynold Cheng [†]
[#]Department of Computer Science, Tsinghua University
[†]Department of Computer Science, The University of Hong Kong
ydzheng2@cs.hku.hk, liguoliang@tsinghua.edu.cn
yb-li16@mails.tsinghua.edu.cn, chshan@cs.hku.hk, ckcheng@cs.hku.hk

ABSTRACT

Crowdsourcing has emerged as a novel problem-solving paradigm, which facilitates addressing problems that are hard for computers, e.g., entity resolution and sentiment analysis. However, due to the openness of crowdsourcing, workers may yield low-quality answers, and a redundancy-based method is widely employed, which first assigns each task to multiple workers and then infers the correct answer (called *truth*) for the task based on the answers of the assigned workers. A fundamental problem in this method is *Truth Inference*, which decides how to effectively infer the truth. Recently, the database community and data mining community independently study this problem and propose various algorithms. However, these algorithms are not compared extensively under the same framework and it is hard for practitioners to select appropriate algorithms. To alleviate this problem, we provide a detailed survey on 17 existing algorithms and perform a comprehensive evaluation using 5 real datasets. We make all codes and datasets public for future research. Through experiments we find that existing algorithms are not stable across different datasets and there is no algorithm that outperforms others consistently. We believe that the truth inference problem is not fully solved, and identify the limitations of existing algorithms and point out promising research directions.

1. INTRODUCTION

Crowdsourcing solutions have been proposed to address tasks that are hard for machines, e.g., entity resolution [8] and sentiment analysis [32]. Due to the wide deployment of public crowdsourcing platforms, e.g., Amazon Mechanical Turk (AMT) [2], CrowdFlower [12], the access to crowd becomes much easier. As reported in [1], more than 500K workers from 190 countries have performed tasks on AMT [2]. The database community has shown great interests in crowdsourcing (see a survey [29]). Several crowdsourced databases (e.g., CrowdDB [20], Deco [39], Qurk [37]) are built to incorporate the crowd into query processing, and there are many studies on implementing crowdsourced operators, e.g., Join [50, 36, 52, 11], Max [47, 22], Top-k [14, 55], Group-by [14], etc.

Due to the openness of crowdsourcing, the crowd (called *workers*) may yield low-quality or even noisy answers. Thus it is important to control the quality in crowdsourcing. To address this problem, most of existing crowdsourcing studies employ a redundancy-based strategy, which assigns each task to multiple workers and aggregates the answers given by different workers to infer the correct answer (called *truth*) of each task. A fundamental problem, called *Truth Inference*, is widely studied in existing crowdsourcing works [34, 16, 15, 53, 51, 41, 26, 33, 61, 19, 35, 30, 27, 10, 46, 5, 31], which decides how to effectively infer the truth for each task.

To address the problem, a straightforward approach is Majority Voting (MV), which takes the answer given by majority workers as the truth. However, the biggest limitation of MV is that it regards all workers as equal. In reality, workers may have different levels of qualities: a high-quality worker carefully answers tasks; a low-quality (or spammer) may randomly answer tasks in order to deceive money; a malicious worker may even intentionally give wrong answers. Thus it is important to capture each worker's quality, which can better infer the truth of each task by trusting more on the answers given by workers with higher qualities.

However, the ground truth of each task is unknown and it is hard to estimate a worker's quality. To address this problem, one can label the ground truth for a small portion of tasks (called *golden tasks*) and use them to estimate workers' quality. There are two types of methods to utilize golden tasks. The first is **qualification test**. Each worker requires to perform a set of golden tasks before she can really answer tasks, and her quality is computed based on her answering performance for these golden tasks. The second is **hidden test**. The golden tasks are mixed into the tasks and the workers do not know which are golden tasks. A worker's quality is computed based on her answering performance on these golden tasks. However, the two approaches have some limitations. (1) For qualification test, workers require to answer these "extra" tasks without pay, and many workers do not want to answer such tasks. (2) For hidden test, it is a waste to pay the "extra" tasks. (3) The two techniques may not improve the quality (see Section 6).

Considering these limitations, the database community [34, 19, 35, 24, 30, 31, 58] and data mining community [16, 53, 15, 61, 27, 46, 41, 51, 26, 33, 5] independently study this problem and propose various algorithms. However, these algorithms are not compared under the same experimental framework and it is hard for practitioners to select appropriate algorithms. To alleviate this problem, we provide a comprehensive survey on existing truth inference algorithms. We summarize them in terms of *task types*, *task modeling*, *worker modeling*, and *inference techniques*. We conduct a comprehensive comparison of 17 existing representative methods [16, 53, 15, 61, 27, 46, 41, 30, 5, 31, 51, 26, 33], experimentally compare them on 5 real datasets with varying sizes and task types in real crowdsourcing platforms, make a deep analysis on the experimental results, and provide extensive experimental findings.

Proceedings of the VLDB Endowment, Vol. 10, No. 5

Table 1: A Product Dataset.

ID	Product Name
r_1	*iPad Two 16GB WiFi White*
r_2	*iPad 2nd generation 16GB WiFi White*
r_3	*Apple iPhone 4 16GB White*
r_4	*iPhone 4th generation White 16GB*

Table 2: Collected Workers' Answers for All Tasks.

	t_1: (r_1=r_2)	t_2: (r_1=r_3)	t_3: (r_1=r_4)	t_4: (r_2=r_3)	t_5: (r_2=r_4)	t_6: (r_3=r_4)
w_1	F	T	T	F	F	F
w_2		F	F	T	T	F
w_3	T	F	F	F	F	T

To summarize, we make the following contributions:

• We survey 17 existing algorithms, summarize a framework (Section 3), and provide an in-depth analysis and summary on the 17 algorithms in different perspectives (Sections 4-5), which can help practitioners to easily grasp existing truth inference algorithms.

• We experimentally conduct a thorough comparison of these methods on 5 datasets with varying sizes, publicize our codes and datasets [40], and provide experimental findings, which give guidance for selecting appropriate methods under various scenarios (Section 6).

• We find that the truth inference problem is not fully solved, identify the limitations of existing algorithms, and point out several promising research directions (Section 7).

2. PROBLEM DEFINITION

DEFINITION 1 (TASK). *A task set $\mathcal{T}$ contains n tasks, i.e., $\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$. Each task asks workers to answer the task.*

Existing studies mainly focus on three types of tasks.

Decision-Making Tasks. A decision-making task has a claim and asks workers to make a decision on whether the claim is *true* (denoted as 'T') or *false* (denoted as 'F'). Decision-making tasks are widely used and studied in existing crowdsourcing works [34, 16, 15, 53, 51, 41, 26, 33, 61, 19, 35, 30, 27, 46, 5] because of its conceptual simplicity.

Next we take entity resolution as an example, which tries to find pairs of products in Table 1 that refer to the same real-world entity. A straightforward way is to generate a task set $\mathcal{T} = \{(r_1$=$r_2), (r_1$=$r_3), (r_1$=$r_4), (r_2$=$r_3), (r_2$=$r_4), (r_3$=$r_4)\}$ with $n = 6$ decision-making tasks, where each task has two choices: (*true*, *false*), and asks workers to select a choice for the task. For example, t_2 (or r_1=r_3) asks whether the claim '*iPad Two 16GB WiFi White = Apple iPhone 4 16GB White*' is *true* ('T') or *false* ('F'). Tasks are then published to crowdsourcing platforms (e.g., AMT [2]) and workers' answers are collected.

Single-Choice (and Multiple-Choice) Tasks. A single-choice task contains a question and a set of candidate choices, and asks workers to select a single choice out of the candidate choices. For example, in sentiment analysis, a task asks workers to select the sentiment ('*positive*', '*neutral*', '*negative*') of a given tweet. Decision-making task is a special case of single-choice task, with two special choices ('T' and 'F'). The single-choice tasks are especially studied in [34, 16, 15, 53, 41, 61, 35, 30, 27, 46, 5]. A direct extension of single-choice task is multiple-choice task, where workers can select multiple choices (not only a single choice) out of a set of candidate choices. For example, in image tagging, given a set of candidate tags for an image, it asks workers to select the tags that the image contains. However, as addressed in [60, 38], a multiple-choice task can be easily transformed to a set of decision-making tasks, e.g., for an image tagging task (multiple-choice), each transformed decision-making task asks whether or not a tag is contained in an image. Thus the methods in decision-making tasks can be directly extended to handle multiple-choice tasks.

Table 3: Notations.

Notation	Description
t_i	the i-th task ($1 \le i \le n$) and $\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$
w	the worker w and $\mathcal{W} = \{w\}$ is the set of workers
$\mathcal{W}^i$	the set of workers that have answered task t_i
$\mathcal{T}^w$	the set of tasks that have been answered by worker w
v_i^w	the answer given by worker w for task t_i
V	the set of workers' answers for all tasks, i.e., $V = \{v_i^w\}$
v_i^*	the (ground) truth for task t_i ($1 \le i \le n$)

Numeric Tasks. The numeric task asks workers to provide a value. For example, a task asks about the height of Mount Everest. Different from the tasks above, workers' inputs are numeric values, which have inherent orderings (e.g., compared with 8800m, 8845m is closer to 8848m). Existing works [41, 30] especially study such tasks by considering the inherent orderings between values.

Others. Besides the above tasks, there are other types of tasks, e.g., translate a language to another [10], or ask workers to collect data (e.g., the name of a celebrity) [20, 48]. However, it is hard to control the quality for such "open" tasks. Thus they are rarely studied in existing works [10, 20, 48]. In this paper, we focus only on the above three tasks and leave other tasks for future work.

DEFINITION 2 (WORKER). *A worker set $\mathcal{W}$ contains a set of workers, i.e., $\mathcal{W} = \{w\}$. Let $\mathcal{W}^i$ denote the set of workers that have answered task t_i and $\mathcal{T}^w$ denote the set of tasks that have been answered by worker w.*

DEFINITION 3 (ANSWER). *Each task t_i can be answered with a subset of workers in $\mathcal{W}$. Let v_i^w denote the worker w's answer for task t_i, and the set of answers $V = \{v_i^w\}$ contains the collected workers' answers for all tasks.*

Table 2 shows an example, with answers to $\mathcal{T}$ given by three workers $\mathcal{W} = \{w_1, w_2, w_3\}$. (The empty cell means that the worker does not answer the task.) For example, $v_4^{w_1}$ = F means worker w_1 answers t_4 (i.e., $r_2 = r_3$) with 'F', i.e., w_1 thinks that $r_2 \neq r_3$. The set of workers that answer t_1 is $\mathcal{W}^1 = \{w_1, w_3\}$, and the set of tasks answered by worker w_2 is $\mathcal{T}^{w_2} = \{t_2, t_3, t_4, t_5, t_6\}$.

DEFINITION 4 (TRUTH). *Each task t_i has a true answer, called the* ground truth *(or* truth*), denoted as v_i^*.*

For the example task set $\mathcal{T}$ in Table 1, only pairs (r_1= r_2) and (r_3= r_4) are *true*, and thus $v_1^* = v_6^*$ = T, and others' truth are F.

Based on the above notations, the truth inference problem is to infer the (unknown) truth v_i^* for each task t_i based on V.

DEFINITION 5 (TRUTH INFERENCE IN CROWDSOURCING). *Given workers' answers V, infer the truth v_i^* of each task $t_i \in \mathcal{T}$.*

Table 3 summarizes the notations used in the paper.

3. SOLUTION FRAMEWORK

A naive solution is Majority Voting (MV) [20, 39, 37], which regards the choice answered by majority workers as the truth. Based on Table 2, the truth derived by MV is v_i^* = F for $2 \le i \le 6$ and it randomly infers v_1^* to break the tie. The MV incorrectly infers v_6^*, and has 50% chance to infer v_1^* wrongly. The reason is that MV assumes that each worker has the same quality, and in reality, workers have different qualities: some are experts or ordinary workers, while others are spammers (who randomly answer tasks in order to deceive money) or even malicious workers (who intentionally give wrong answers). Take a closer look at Table 2, we can observe that w_3 has a higher quality, and the reason is that if we do not consider t_1 (which receives 1 'T' and 1 'F'), then w_3 gives 4 out of 5 answers that are reported by majority workers, while w_1 and w_2 give both 3 out of 5, thus we should give higher trust to w_3's answer and in this way can infer all tasks' truth correctly.

Based on the above discussions, existing works [16, 15, 53, 51, 41, 33, 26, 61, 62, 19, 35, 30, 46, 27, 5, 34] propose various ways to model a worker's quality. Although qualification test and hidden test can help to estimate a worker's quality, they require to label tasks with truth beforehand, and a worker also requires to answer these "extra" tasks. To address this problem, existing works [16, 15, 53, 51, 41, 33, 26, 61, 62, 19, 35, 30, 46, 27, 5, 34] estimate each worker's quality purely based on workers' answers V. Intuitively, they capture the inherent relations between workers' qualities and tasks' truth: for a task, the answer given by a high-quality worker is highly likely to be the truth; conversely, for a worker, if the worker often correctly answers tasks, then the worker will be assigned with a high quality. By capturing such relations, they adopt an iterative approach, which jointly infers both the workers' qualities and tasks' truth.

By capturing the above relations, the general approach adopted by most of existing works [16, 15, 53, 51, 41, 33, 26, 61, 62, 19, 35, 30, 46, 27, 5, 34] is shown in Algorithm 1. The quality of each worker $w \in \mathcal{W}$ is denoted as q^w. In Algorithm 1, it first initializes workers' qualities randomly or using qualification test (line 1), and then adopts an iterative approach with two steps (lines 3-11):

Step 1: Inferring the Truth (lines 3-5): it infers each task's truth based on workers' answers and qualities. In this step, different task types are handled differently. Furthermore, some existing works [53, 51] explicitly model each task, e.g., [53] regards that different tasks may have different difficulties. We discuss how existing works model a task in Section 4.1.

Step 2: Estimating Worker Quality (lines 6-8): based on workers' answers and each task's truth (derived from step 1), it estimates each worker's quality. In this step, existing works model each worker w's quality q^w differently. For example, [16, 26, 33, 5] model q^w as a single value, while [15, 41, 33, 27, 46] model q^w as a matrix. We discuss worker's models in Section 4.2.

Convergence (lines 9-11): the two iterations will run until convergence. Typically to identify convergence, existing works will check whether the change of two sets of parameters (i.e., workers' qualities and tasks' truth) is below some defined threshold (e.g., 10^{-3}). Finally the inferred truth and workers' qualities are returned.

Running Example. Let us show how the method PM [31, 5] works for Table 2. PM models each worker w as a single value $q^w \in [0, +\infty)$ and a higher value implies a higher quality. Initially, each worker $w \in \mathcal{W}$ is assigned with the same quality $q^w = 1$. Then the two steps devised in PM are as follows:

Step 1 (line 5): $v_i^* = \text{argmax}_v \sum_{w \in \mathcal{W}^i} q^w \cdot \mathbb{1}_{\{v = v_i^w\}}$;

Step 2 (line 8): $q^w = -\log\big(\frac{\sum_{t_i \in \mathcal{T}^w} \mathbb{1}_{\{v_i^* \neq v_i^w\}}}{\max_{w \in \mathcal{W}}\{\sum_{t_i \in \mathcal{T}^w} \mathbb{1}_{\{v_i^* \neq v_i^w\}}\}}\big)$.

The indicator function $\mathbb{1}_{\{\cdot\}}$ returns 1 if the statement is true; 0, otherwise. For example, $\mathbb{1}_{\{5=3\}} = 0$ and $\mathbb{1}_{\{5=5\}} = 1$. For the 1st iteration, in step 1, it computes each task's truth from workers' answers by considering which choice receives the highest aggregated workers' qualities. Intuitively, the answer given by many high quality workers are likely to be the truth. For example, for task t_2, as it receives one T and two F's from workers and each worker is of the same quality, then v_2^* = F. Similarly we get v_1^* = T and v_i^* = F for $2 \leq i \leq 6$. In step 2, based on the computed truth in step 1, it gives a high (low) quality to a worker if the worker makes few (a lot of) mistakes. For example, as the number of mistakes (i.e., $\sum_{t_i \in \mathcal{T}^w} \mathbb{1}_{\{v_i^* \neq v_i^w\}}$) for workers w_1, w_2, w_3 are 3, 2, 1, respectively, thus the computed qualities are $q^{w_1} = -\log(3/3) = 0$, $q^{w_2} = -\log(2/3) = 0.41$ and $q^{w_3} = -\log(1/3) = 1.10$. Following these two steps, the process will then iterate until convergence. In the converged results, the truth are $v_1^* = v_6^*$ = T, and v_i^* = F ($2 \leq i \leq 5$); the qualities are $q^{w_1} = 4.9 \times 10^{-15}$, $q^{w_2} = 0.29$ and $q^{w_3} = 16.09$. We can observe that PM can derive the truth correctly, and w_3 has a higher quality compared with w_1 and w_2.

Algorithm 1: *Solution Framework*

Input: workers' answers V
Output: inferred truth v_i^* ($1 \leq i \leq n$), worker quality q^w ($w \in \mathcal{W}$)

```
1  Initialize all workers' qualities (q^w for w ∈ W);
2  while true do
3      // Step 1: Inferring the Truth
4      for 1 ≤ i ≤ n do
5          Inferring the truth v_i^* based on V and {q^w | w ∈ W};
6      // Step 2: Estimating Worker Quality
7      for w ∈ W do
8          Estimating the quality q^w based on V and {v_i^* | 1 ≤ i ≤ n};
9      // Check for Convergence
10     if Converged then
11         break;
12 return v_i^* for 1 ≤ i ≤ n and q^w for w ∈ W;
```

4. IMPORTANT FACTORS

In this section, we categorize existing works [16, 15, 53, 51, 41, 33, 26, 61, 62, 19, 35, 30, 46, 27, 5, 34] following two factors:

Task Modeling (Section 4.1): how existing works model a task (e.g., task's difficulty, latent topics).

Worker Modeling (Section 4.2): how existing works model a worker's quality (e.g., worker probability, diverse skills).

We summarize how existing works [16, 15, 53, 51, 41, 33, 26, 61, 62, 19, 35, 30, 46, 27, 5, 34] can be categorized based on the above factors in Table 4. Next we analyze each factor, respectively.

4.1 Task Modeling

4.1.1 Task Difficulty

Different from most existing works which assume that a worker has the same quality for answering different tasks, some recent works [53, 35] model the difficulty in each task. They assume that each task has its difficulty level, and the more difficult a task is, the harder a worker can correctly answer the task. For example, in [53], it models the probability that worker w correctly answers task t_i as follows: $\Pr(v_i^w = v_i^* \mid d_i, q^w) = 1/(1 + e^{-d_i \cdot q^w})$, where $d_i \in (0, +\infty)$ represents the difficulty for task t_i, and the higher d_i is, the easier task t_i is. Intuitively, for a fixed worker quality $q^w > 0$, an easier task (high value of d_i) leads to a higher probability that the worker correctly answers the task.

4.1.2 Latent Topics

Different from modeling each task as a value (e.g., difficulty), some recent works [19, 35, 57, 51] model each task as a vector with K values. The basic idea is to exploit the diverse topics in a task, where the topic number (i.e., K) is pre-defined. For example, existing studies [19, 35] make use of the text description in each task and adopt topic model techniques [6, 56] to generate a vector of size K for the task; while Multi [51] learns a K-size vector without referring to external information (e.g., text descriptions). Based on the task models, a worker is probable to answer a task correctly if the worker has high qualities on the task's related topics.

4.2 Worker Modeling

4.2.1 Worker Probability

Worker probability uses a single real number (between 0 and 1) to model a worker w's quality $q^w \in [0, 1]$, which represents the ability that worker w correctly answers a task. The higher q^w is, the worker w has higher ability to correctly answer tasks. The model

Table 4: Comparisons of Different Methods that Address Truth Inference Problem in Crowdsourcing.

Method	Task Types	Task Modeling	Worker Modeling	Techniques
MV	Decision-Making, Single-Choice	No Model	No Model	Direct Computation
ZC [16]	Decision-Making, Single-Choice	No Model	Worker Probability	Probabilistic Graphical Model
GLAD [53]	Decision-Making, Single-Choice	Task Difficulty	Worker Probability	Probabilistic Graphical Model
D&S [15]	Decision-Making, Single-Choice	No Model	Confusion Matrix	Probabilistic Graphical Model
Minimax [61]	Decision-Making, Single-Choice	No Model	Diverse Skills	Optimization
BCC [27]	Decision-Making, Single-Choice	No Model	Confusion Matrix	Probabilistic Graphical Model
CBCC [46]	Decision-Making, Single-Choice	No Model	Confusion Matrix	Probabilistic Graphical Model
LFC [41]	Decision-Making, Single-Choice	No Model	Confusion Matrix	Probabilistic Graphical Model
CATD [30]	Decision-Making, Single-Choice, Numeric	No Model	Worker Probability, Confidence	Optimization
PM [5, 31]	Decision-Making, Single-Choice, Numeric	No Model	Worker Probability	Optimization
Multi [51]	Decision-Making	Latent Topics	Diverse Skills, Worker Bias, Worker Variance	Probabilistic Graphical Model
KOS [26]	Decision-Making	No Model	Worker Probability	Probabilistic Graphical Model
VI-BP [33]	Decision-Making	No Model	Confusion Matrix	Probabilistic Graphical Model
VI-MF [33]	Decision-Making	No Model	Confusion Matrix	Probabilistic Graphical Model
LFC_N [41]	Numeric	No Model	Worker Variance	Probabilistic Graphical Model
Mean	Numeric	No Model	No Model	Direct Computation
Median	Numeric	No Model	No Model	Direct Computation

has been widely used in existing works [16, 26, 33, 5]. Some recent works [53, 31] extend the worker probability to model a worker's quality in a wider range, e.g., $q^w \in (-\infty, +\infty)$, and a higher q^w means the worker w's higher quality in answering tasks.

4.2.2 Confusion Matrix

Confusion matrix [15, 41, 33, 27, 46] is used to model a worker's quality for answering single-choice tasks. Suppose each task in $\mathcal{T}$ has ℓ fixed choices, then the confusion matrix q^w is an $\ell \times \ell$ matrix, where the j-th ($1 \le j \le \ell$) row, i.e., $q^w_{j,\cdot} = [\, q^w_{j,1}, q^w_{j,2}, \ldots, q^w_{j,\ell}\,]$, represents the probability distribution of worker w's possible answers for a task if the truth of the task is the j-th choice. Each element $q^w_{j,k}$ ($1 \le j \le \ell, 1 \le k \le \ell$) means that "given the truth of a task is the j-th choice, the probability that worker w selects the k-th choice", i.e., $q^w_{j,k} = \Pr(v^w_i = k \mid v^*_i = j)$ for any $t_i \in \mathcal{T}$. For example, decision-making tasks ask workers to select 'T' (1st choice) or 'F' (2nd choice) for each claim ($\ell = 2$), then an example confusion matrix for w is $q^w = \begin{bmatrix} 0.8 & 0.2 \\ 0.3 & 0.7 \end{bmatrix}$, where $q^w_{1,2} = 0.2$ means that if the truth of a task is 'T', the probability that the worker answers the task as 'F' is 0.2.

4.2.3 Worker Bias and Worker Variance

Worker bias and variance [51, 41] are proposed to handle numeric tasks, where worker bias captures the effect that a worker may underestimate (or overestimate) the truth of a task, and worker variance captures the variation of errors around the bias. For example, given a set of photos with humans, each numeric task asks workers to estimate the height of the human on it. Suppose a worker w is modeled with bias τ_w and variance σ_w, then the answer v^w_i given by worker w is modeled to draw from the Gaussian distribution: $v^w_i \sim \mathcal{N}(v^*_i + \tau_w, \sigma_w)$, that is, (1) a worker with bias $\tau_w \gg 0$ ($\tau_w \ll 0$) will overestimate (underestimate) the height, while $\tau_w \to 0$ leads to more accurate estimation; (2) a worker with variance $\sigma_w \gg 0$ means a large variation of error, while $\sigma_w \to 0$ leads to a small variation of error.

4.2.4 Confidence

Existing works [30, 25] observe that if a worker answers plenty of tasks, then the estimated quality for the worker is confident; otherwise, if a worker answers only a few tasks, then the estimated quality is not confident. Inspired by this observation, [35] assigns higher qualities to the workers who answer plenty of tasks, than the workers who answer a few tasks. To be specific, for a worker w, it uses the Chi-Square distribution [3] with 95% confidence interval, i.e., $\mathcal{X}^2_{(0.975,|\mathcal{T}^w|)}$ as a coefficient to scale up the worker's quality, where $|\mathcal{T}^w|$ is the number of tasks that worker w has answered. $\mathcal{X}^2_{(0.975,|\mathcal{T}^w|)}$ increases with $|\mathcal{T}^w|$, i.e., the more tasks w has answered, the higher worker w's quality is scaled to.

4.2.5 Diverse Skills

A worker may have various levels of expertise for different topics. For example, a sports fan that rarely pays attention to entertainment may answer tasks related to *sports* more correctly than tasks related to *entertainment*. Different from most of the above models which have an assumption that a worker has the *same* quality to answer different tasks, existing works [19, 35, 61, 51, 57, 59] model the diverse skills in a worker and capture a worker's diverse qualities for different tasks. The basic ideas of [19, 61] are that they model a worker w's quality as a vector of size n, i.e., $q^w = [\, q^w_1, q^w_2, \ldots, q^w_n\,]$, where q^w_i indicates worker w's quality for task t_i. Different from [19, 61], some recent works [35, 51, 57, 59] model a worker's quality for different latent topics, i.e., $q^w = [\, q^w_1, q^w_2, \ldots, q^w_K\,]$, where the number K is pre-defined, indicating the number of latent topics. They [35, 51, 57, 59] assume that each task is related to one or more topics in these K latent topics, and a worker is highly probable to correctly answer a task if the worker has a high quality in the task's related topics.

5. TRUTH INFERENCE ALGORITHMS

Existing works [61, 19, 30, 5, 34, 16, 15, 53, 51, 41, 26, 33, 35, 27, 46] usually adopt the framework in Algorithm 1. Based on the used techniques, they can be classified into the following three categories: direct computation [20, 39], optimization methods [61, 19, 30, 5] and probabilistic graphical model methods [34, 16, 15, 53, 51, 41, 26, 33, 35, 27, 46]. Next we talk about them, respectively.

5.1 Direct Computation

Some baseline methods directly estimate v^*_i ($1 \le i \le n$) based on V, without modeling each worker or task. For decision-making and single-label tasks, Majority Voting (MV) regards the truth of each task as the answer given by most workers; while for numeric tasks, Mean and Median are two baseline methods that regard the mean and median of workers' answers as the truth for each task.

5.2 Optimization

The basic idea of optimization methods is to set a self-defined optimization function that captures the relations between workers' qualities and tasks' truth, and then derive an iterative method to compute these two sets of parameters collectively. The differences among existing works [5, 31, 30, 61] are that they model workers' qualities differently and apply different optimization functions to capture the relations between the two sets of parameters.

(1) Worker Probability. PM [5, 31] models each worker's quality as a single value, and the optimization function is defined as:

$$\min_{\{q^w\},\{v_i^*\}} f(\{q^w\},\{v_i^*\}) = \sum_{w\in\mathcal{W}} q^w \cdot \sum_{t_i\in\mathcal{T}^w} d(v_i^w, v_i^*),$$

where $\{q^w\}$ represents the set of all workers' qualities, and similarly $\{v_i^*\}$ represents the set of all truth. It models a worker w's quality as $q^w \geq 0$, and $d(v_i^w, v_i^*) \geq 0$ defines the distance between worker's answer v_i^w and the truth v_i^*: the similar v_i^w is to v_i^*, the lower the value of $d(v_i^w, v_i^*)$ is. Intuitively, to minimize $f(\{q^w\},\{v_i^*\})$, a worker w's high quality q^w corresponds to a low value in $d(v_i^*, v_i^w)$, i.e., worker w's answer should be close to the truth. By capturing the intuitions, similar to Algorithm 1, PM [5, 31] develops an iterative approach, and in each iteration, it adopts the two steps as illustrated in Section 3.

(2) Worker Probability and Confidence. Different from above, CATD [30] considers both worker probability and confidence in modeling a worker's quality. As discussed in Section 4.2.4, each worker w's quality is scaled up to a coefficient of $\mathcal{X}^2_{(0.975,|\mathcal{T}^w|)}$, i.e., the more tasks w has answered, the higher worker w's quality is scaled to. It develops an objective function, with the intuitions that a worker w who gives answers close to the truth and answers a plenty of tasks should have a high quality q^w. Similarly it adopts an iterative approach, and iterates the two steps until convergence.

(3) Diverse Skills. Minimax [61] leverages the idea of minimax entropy [63]. To be specific, it models the diverse skills of a worker w across different tasks and focuses on single-label tasks (with ℓ choices). It assumes that for a task t_i, the answers given by w are generated by a probability distribution $\pi_{i,\cdot}^w = [\,\pi_{i,1}^w, \pi_{i,2}^w, \ldots, \pi_{i,\ell}^w\,]$, where each $\pi_{i,j}^w$ is the probability that worker w answers task t_i with the j-th choice. Following this, an objective function is defined by considering two constraints for tasks and workers: for a task t_i, the number of answers collected for a choice equals the sum of corresponding generated probabilities; for a worker w, among all tasks answered by w, given the truth is the j-th choice, the number of answers collected for the k-th choice equals the sum of corresponding generated probabilities. Finally [61] devises an iterative approach to infer the two sets of parameters $\{v_i^*\}$ and $\{\pi^w\}$.

5.3 Probabilistic Graphical Model (PGM)

A probabilistic graphical model (PGM) [28] is a graph which expresses the conditional dependency structure (represented by edges) between random variables (represented by nodes). Figure 1 shows the general PGM adopted in existing works. Each node represents a variable. There are two plates, respectively for workers and tasks, where each one represents the repeating variables. For example, the plate for workers represents $|\mathcal{W}|$ repeating variables, where each variable corresponds to a worker $w \in \mathcal{W}$. For the variables, α, β, and v_i^w are known (α and β are priors for q^w and v_i^*, which can be set based on the prior knowledge); q^w and v_i^* are latent or unknown variables, which are two desired variables to compute. The directed edges model the conditional dependence between a child node and its associated parent node(s) in the sense that the child node follows a probabilistic distribution conditioned on the values taken by the parent node(s). For example, three conditional distributions in Figure 1 are $\Pr(q^w \mid \alpha)$, $\Pr(v_i^* \mid \beta)$ and $\Pr(v_i^w \mid q^w, v_i^*)$.

Next we illustrate the details (optimization goal and the two steps) of each method using PGM. In general the methods differ in the used worker model. It can be classified into three categories: **worker probability** [16, 53, 26, 33], **confusion matrix** [15, 41, 27, 46] and **diverse skills** [19, 35, 51]. For each category, we first introduce its basic method, e.g., ZC [16], and then summarize how other methods [53, 26, 33] extend the basic method ZC [16].

(1) Worker Probability: ZC [16] and its extensions [53, 26, 33].

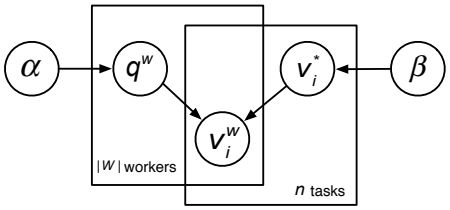

Figure 1: A General PGM (Probabilistic Graphical Model).

ZC [16] adopts a PGM similar to Figure 1, with the simplification that it does not consider the priors (i.e., α, β). Suppose all tasks are decision-making tasks ($v_i^* \in \{\mathrm{T}, \mathrm{F}\}$) and each worker's quality is modeled as worker probability $q^w \in [0,1]$. Then

$$\Pr(v_i^w \mid q^w, v_i^*) = (q^w)^{\mathbb{1}_{\{v_i^w = v_i^*\}}} \cdot (1-q^w)^{\mathbb{1}_{\{v_i^w \neq v_i^*\}}},$$

which means that the probability worker w correctly (incorrectly) answers a task is q^w ($1-q^w$). For decision-making tasks, ZC [16] tries to maximize the probability of the occurrence of workers' answers, called *likelihood*, i.e., $\max_{\{q^w\}} \Pr(V \mid \{q^w\})$, which regards $\{v_i^*\}$ as latent variables:

$$\Pr(V \mid \{q^w\}) = \frac{1}{2} \cdot \prod_{i=1}^{n} \sum\nolimits_{z\in\{\mathrm{T},\mathrm{F}\}} \prod\nolimits_{w\in\mathcal{W}^i} \Pr(v_i^w \mid q^w, v_i^* = z). \quad (1)$$

However, it is hard to optimize due to the non-convexity. Thus ZC [16] applies the EM (Expectation-Maximization) framework [17] and iteratively updates q^w and v_i^* to approximate its optimal value. Note ZC [16] develops a system to address entity linking for online pages. In this paper we focus on the part of leveraging the crowd's answers to infer the truth (i.e., Section 4.3 in [16]), and we omit other parts (e.g., constraints on its probabilistic model).

There are several extensions of ZC, e.g., GLAD [53], KOS [26], VI-BP [33], VI-MF [33], and they focus on different perspectives:

Task Model. GLAD [53] extends ZC [16] in task model. Rather than assuming that each task is the same, it [53] models each task t_i's difficulty $d_i \in (0, +\infty)$ (the higher, the easier). Then it models the worker's answer as $\Pr(v_i^w = v_i^* \mid d_i, q^w) = 1/(1+e^{-d_i \cdot q^w})$, and integrates it into Equation 1 to approximate the optimal value using Gradient Descent [28] (an iterative method).

Optimization Function. KOS [26], VI-BP [33], and VI-MF [33] extend ZC [16] in an optimization goal. Recall that ZC tries to compute the optimal $\{q^w\}$ that maximizes $\Pr(V \mid \{q^w\})$, which is the *Point Estimate*. Instead, [26, 33] leverage the *Bayesian Estimators* to calculate the integral of all possible q^w, and the target is to estimate the truth $v_i^* = \mathrm{argmax}_{z\in\{\mathrm{T},\mathrm{F}\}} \Pr(v_i^* = z \mid V)$, where

$$\Pr(v_i^* = z \mid V) = \int_{\{q^w\}} \Pr(v_i^* = z, \{q^w\} \mid V)\ \mathrm{d}\{q^w\}. \quad (2)$$

It is hard to directly compute Equation 2, and existing works [26, 33] seek for *Variational Inference* (*VI*) techniques [49] to approximate the value: KOS [26] first leverages *Belief Propagation* (one typical *VI* technique) to iteratively approximate the value in Equation 2, then [33] proposes a more general model based on KOS, called VI-BP. Moreover, it [33] also applies *Mean Field* (anther *VI* technique) in VI-MF to iteratively approach Equation 2.

(2) Confusion Matrix: D&S [15] and its extensions [41, 27, 46].

D&S [15] focuses on single-label tasks (with fixed ℓ choices) and models each worker as a confusion matrix q^w with size $\ell \times \ell$ (Section 4.2.2). The worker w's answer follows the probability $\Pr(v_i^w \mid q^w, v_i^*) = q^w_{v_i^*, v_i^w}$. Similar to Equation 1, D&S [15] tries to optimize the function $\mathrm{argmax}_{\{q^w\}} \Pr(V \mid \{q^w\})$, where

$$\Pr(V \mid \{q^w\}) = \prod\nolimits_{i=1}^{n} \sum\nolimits_{1\leq z\leq \ell} \Pr(v_i^* = z) \cdot \prod\nolimits_{w\in\mathcal{W}^i} q^w_{z, v_i^w},$$

and it applies the EM framework [17] to devise two iterative steps.

The above method D&S [15], which models a worker as a confusion matrix, is also a widely used model. There are some extensions, e.g., LFC [41], LFC_N [41], BCC [27] and CBCC [46].

Table 5: The Statistics of Each Dataset.

Dataset	#tasks (n)	#truth	$\|V\|$	$\|V\|/n$	$\|\mathcal{W}\|$
Datasets for Decision-Making Tasks					
D_Product [50]	8,315	8,315	24,945	3	176
D_PosSent	1,000	1,000	20,000	20	85
Datasets for Single-Label Tasks					
S_Rel [9]	20,232	4,460	98,453	4.9	766
S_Adult [4]	11,040	1,517	92,721	8.4	825
Datasets for Numeric Tasks					
N_Emotion [44]	700	700	7,000	10	38

Priors. LFC [41] extends D&S [15] to incorporate the priors into worker's model, by assuming that the priors, denoted as $\alpha^w_{j,k}$ for $1 \le j,k \le \ell$ are known in advance, and the worker's quality $q^w_{j,k}$ is generated following Beta$(\alpha^w_{j,k}, \sum_{k=1}^{\ell} \alpha^w_{j,k})$ distribution.

Task Type. LFC_N [41] also handles numeric tasks. Different from decision-making and single-choice tasks, it assumes that worker w's answer follows $v^w_i \sim \mathcal{N}(v^*_i, \sigma^2_w)$, where σ_w is the variance, and a small σ_w implies that v^w_i is close to truth v^*_i.

Optimization Function. BCC [27] has a different optimization goal compared with D&S [15] and it aims at maximizing the posterior joint probability. For example, in Figure 1, it optimizes the posterior joint probability of all unknown variables, i.e.,

$$\prod_{i=1}^{n} \Pr(v^*_i \mid \beta) \prod_{w \in \mathcal{W}} \Pr(q^w \mid \alpha) \prod_{i=1}^{n} \prod_{w \in \mathcal{W}^i} \Pr(v^w_i \mid q^w, v^*_i).$$

To optimize the above formula, the technique of Gibbs Sampling [28] is used to iteratively infer the two sets of parameters $\{q^w\}, \{v^*_i\}$ until convergence, where q^w is modeled as a confusion matrix. Then CBCC [46] extends BCC [27] to support community. The basic idea is that each worker belongs to one community, where each community has a representative confusion matrix, and workers in the same community share very similar confusion matrices.

(3) Diverse Skills: Multi [51] and others [19, 35, 59].

Recently, there are some works (e.g., [51, 19, 35, 59]) that model a worker's diverse skills. Basically, they model a worker w's quality q^w as a vector of size K (Section 4.2.5), which captures a worker's diverse skills over K latent topics. For example, [35] combines the process of topic model (i.e., TwitterLDA [56]) and truth inference together, and [59] leverages entity linking and knowledge base to exploit a worker's diverse skills.

6. EXPERIMENTS

In this section, we evaluate 17 existing methods (Table 4) on real datasets. We first introduce the experimental setup (Section 6.1), and then analyze the quality of collected crowdsourced data (Section 6.2). Finally we compare with existing methods (Section 6.3). We have made all our used datasets and codes available [40] for reproducibility and future research. We implement the experiments in Python on a server with CPU 2.40GHz and 60GB memory.

6.1 Experimental Setup

6.1.1 Datasets

There are many public crowdsourcing datasets [13]. Among them, we select 5 representative datasets based on three criteria: (1) the dataset is large in task size; (2) each task received multiple answers; (3) all datasets cover different task types. In Table 5, for each selected dataset, we list four statistics: the number of tasks, or #tasks (n), #collected answers ($|V|$), the average number of answers for each task ($|V|/n$), #truth (some large datasets only provide a subset as ground truth) and #workers ($|\mathcal{W}|$). For example, for dataset D_Product, it contains 8,315 tasks, with 24,945 answers collected from 176 workers, and each task is answered with 3 times on average. Next, we introduce the details of each dataset (with different task types). We manually collect answers for D_PosSent [45] from AMT [2]; while for other datasets, we use the public datasets collected by other researchers [50, 9, 4, 44].

Decision-Making Tasks (start with prefix 'D_'):

• D_Product [50]. Each task in the dataset contains two products (with descriptions) and two choices (T, F), and it asks workers to identify whether the claim "*the two products are the same*" is true ('T') or false ('F'). An example task is "*Sony Camera Carrying-LCSMX100* and *Sony LCS-MX100 Camcorder* are the same?". There are 8135 tasks, and 1101 (7034) tasks' truth are T (F).

• D_PosSent. Each task in the dataset contains a tweet related to a company (e.g., "*The recent products of Apple is amazing!*"), and asks workers to identify whether the tweet has positive sentiment to that company. The workers give '*yes*' or '*no*' to each task. Based on the dataset [45], we create 1000 tasks. Among them, 528 (472) tasks' truth are *yes* (*no*). In AMT [2], we batch 20 tasks in a Human Intelligence Task (HIT) and assign each HIT to 20 workers. We pay each worker \$0.03 upon answering a HIT. We manually create qualification test by selecting 20 tasks, and each worker should answer the qualification test before she can answer our tasks.

Single-Choice Tasks (start with prefix 'S_'):

• S_Rel [9]. Each task contains a topic and a document, and it asks workers to choose the relevance of the topic w.r.t. the document by selecting one out of four choices: '*highly relevant*', '*relevant*', '*non-relevant*', and '*broken link*'.

• S_Adult [4]. Each task contains a website, and it asks workers to identify the adult level of the website by selecting one out of four choices: '*G*' (General Audience), '*PG*' (Parental Guidance), '*R*' (Restricted), and '*X*' (Porn).

Numeric Tasks (start with prefix 'N_'):

• N_Emotion [44]. Each task in the dataset contains a text and a range $[-100, 100]$, and it asks each worker to select a score in the range, indicating the degree of emotion (e.g., anger) of the text. A higher score means a higher degree for the emotion.

6.1.2 Metrics

We use different metrics for different task types.

Decision-Making Tasks. We use *Accuracy* as the metric, which is defined as the fraction of tasks whose truth are inferred correctly. Given a method, let $\widehat{v}^*_i$ denote the inferred truth of task t_i, then

$$\textit{Accuracy} = \sum_{i=1}^{n} \mathbb{1}_{\{\widehat{v}^*_i = v^*_i\}}/n. \quad (3)$$

However, for applications such as entity resolution (e.g., dataset D_Product), where the number of F is much larger than the number of T as truth (the proportion of tasks with T and F as truth is 0.12:0.88 in D_Product). In this case, even a naive method that returns all tasks as F achieves very high *Accuracy* (88%), which is not expected, as we care more for the *same entities* (i.e., choice T) in entity resolution. Thus a typical metric *F1-score* is often used, which is defined as the harmonic mean of *Precision* and *Recall*:

$$\textit{F1-score} = \frac{2}{\frac{1}{\textit{Precision}} + \frac{1}{\textit{Recall}}} = \frac{2 \cdot \sum_{i=1}^{n} \mathbb{1}_{\{v^*_i = \mathrm{T}\}} \cdot \mathbb{1}_{\{\widehat{v}^*_i = \mathrm{T}\}}}{\sum_{i=1}^{n} (\mathbb{1}_{\{v^*_i = \mathrm{T}\}} + \mathbb{1}_{\{\widehat{v}^*_i = \mathrm{T}\}})}. \quad (4)$$

Single-Choice Tasks. We use the metric *Accuracy* (Equation 3).

Numeric Tasks. We use two metrics, *MAE* (Mean Absolute Error) and *RMSE* (Root Mean Square Error), defined as below:

$$\textit{MAE} = \frac{\sum_{i=1}^{n} |v^*_i - \widehat{v}^*_i|}{n}, \quad \textit{RMSE} = \sqrt{\frac{\sum_{i=1}^{n} (v^*_i - \widehat{v}^*_i)^2}{n}}, \quad (5)$$

where *RMSE* gives a higher penalty for large errors.

Note that for the metrics *Accuracy* and *F1-score*, they are in $[0, 1]$ and the higher, the better; however, for *MAE* and *RMSE* (defined on errors), they are in $[0, +\infty]$ and the lower, the better.

6.2 Crowdsourced Data Quality

In this section we first ask the following three questions related to the quality of crowdsourced data, and then answer them.

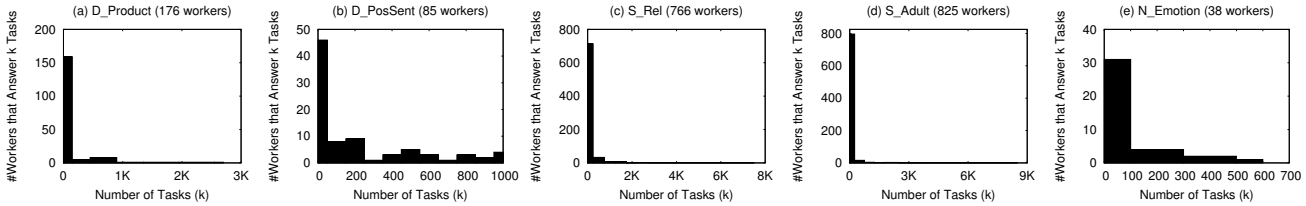

Figure 2: The Statistics of Worker Redundancy for Each Dataset (Section 6.2.2).

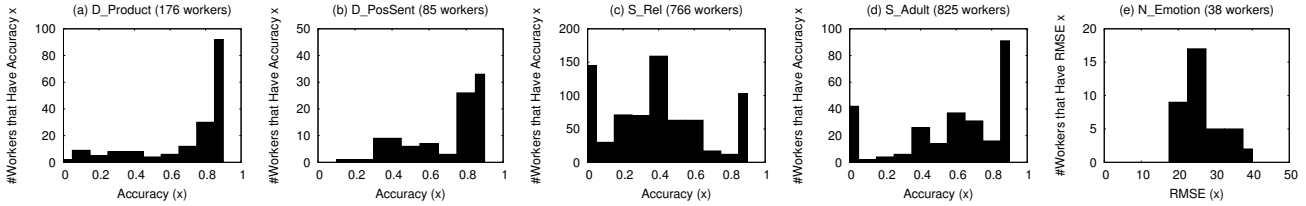

Figure 3: The Statistics of Worker Quality for Each Dataset (Section 6.2.3).

1. Are the crowdsourced data consistent? In other words, are the answers from different workers the same for a task? (Section 6.2.1)
2. Are there a lot of redundant workers? In other words, does each worker answer plenty of tasks? (Section 6.2.2)
3. Do workers provide high-quality data? In other words, are each worker's answers consistent with the truth? (Section 6.2.3)

6.2.1 Data Consistency

Decision-Making & Single-Label Tasks. Note that each task contains a fixed number (denoted as ℓ) of choices. For a task t_i, let $n_{i,j}$ denote the number of answers given to the j-th choice, e.g., in Table 2, for t_2, $n_{2,1} = 1$ and $n_{2,2} = 2$. In order to capture how concentrated the workers' answers are, we first compute the entropy [42] over the distribution of each task's collected answers, and then define data consistency ($\mathcal{C}$) as the average entropy, i.e., $\mathcal{C} = -\frac{1}{n} \cdot \sum_{i=1}^{n} \sum_{j=1}^{\ell} \frac{n_{i,j}}{\sum_{j=1}^{\ell} n_{i,j}} \cdot \log_{\ell} \frac{n_{i,j}}{\sum_{j=1}^{\ell} n_{i,j}}$. Note that we use "$\log_{\ell}$" other than "ln" to ensure that the value $\mathcal{C} \in [0, 1]$, and the lower $\mathcal{C}$ is, the more consistent workers' answers are.

Based on V, we compute $\mathcal{C}$ for each dataset. The computed $\mathcal{C}$ of the four datasets are 0.38, 0.85, 0.82, and 0.39, respectively. It can be seen that the crowdsourced data is not consistent. To be specific, for decision-making and single-label datasets, $\mathcal{C} \geq 0.38$, and there exists highly inconsistent dataset `D_PosSent` with $\mathcal{C} = 0.85$.

Numeric Tasks. As the answers obtained for each task has inherent orderings, in order to capture the consistency of workers' answers, for a task t_i, we first compute the *median* v_i (a robust metric in statistics and it is not sensitive to outliers) over all its collected answers; then the consistency ($\mathcal{C}$) is defined as the average deviation compared with the median, i.e., $\mathcal{C} = \frac{1}{n} \cdot \sum_{i=1}^{n} \sqrt{\frac{\sum_{w \in \mathcal{W}^i} (v_i^w - v_i)^2}{|\mathcal{W}^i|}}$, where $\mathcal{W}^i$ is the set of workers that have answered t_i. We have $\mathcal{C} \in [0, +\infty]$, and a lower $\mathcal{C}$ leads to more consistent answers.

For numeric dataset `N_Emotion`, the computed $\mathcal{C}$ is 20.44.

Summary. The crowdsourced data is inconsistent, which motivates to develop methods that can solve truth inference in crowdsourcing.

6.2.2 Worker Redundancy

For each worker, we define her redundancy as the number of tasks answered by the worker. We record the redundancy of each worker in each dataset, and then draw the histograms of worker redundancies in Figure 2. Specifically, in each dataset, we vary the number of tasks (k), and record how many workers that answer k tasks. We can see in Figure 2 that the worker redundancy conforms to the long-tail phenomenon, i.e., most workers answer a few tasks and only a few workers answer plenty of tasks.

Summary. The worker redundancy of crowdsourced data in real crowdsourcing platforms conforms to long-tail phenomenon.

6.2.3 Worker Quality

In Figure 3, for each dataset, we show each worker's quality, computed based on comparing worker's answers with tasks' truth.

Decision-Making & Single-Label Tasks. We compute each worker w's *Accuracy*, i.e., the proportion of tasks that are correctly answered by w, i.e., $\frac{\sum_{t_i \in \mathcal{T}^w} \mathbb{1}_{\{v_i^w = v_i^*\}}}{|\mathcal{T}^w|} \in [0, 1]$ and a higher value means a higher quality. For each dataset, we compute the corresponding *Accuracy* for each worker and draw the histograms of each worker. It can be seen from Figures 3(a)-(d) that histograms of workers' *Accuracy* are in different shapes for different datasets. To be specific, workers for `D_Product` and `D_PosSent` are of high *Accuracy*, while workers have mediate *Accuracy* for `S_Adult`, and low *Accuracy* for `S_Rel`. The average *Accuracy* for all workers in each dataset are 0.79, 0.79, 0.53 and 0.65, respectively.

Numeric Tasks. It can be seen from Figure 3(e) that workers' *RMSE* vary in $[20, 45]$, and the average *RMSE* is 28.9.

Summary. The workers' qualities vary in the same dataset, which makes it necessary to identify the trustworthy workers.

6.3 Crowdsourced Truth Inference

In this section we compare the performance of existing methods [34, 16, 15, 53, 51, 41, 26, 33, 61, 30, 27, 46, 31, 5]. Our comparisons are performed based on the following perspectives:

1. What is the performance of different methods? In other words, if we only know the workers' answers (i.e., V), which method performs the best? Furthermore, for a method, how does the truth inference quality change with more workers' answers? (Section 6.3.1)
2. What is the effect of qualification test? In other words, if we assume a worker has performed some golden tasks before answering real tasks, and initialize the worker's quality (line 1 in Algorithm 1) based on the worker's answering performance for golden tasks, will this increase the quality of each method? (Section 6.3.2)
3. What is the effect of hidden test? In other words, if we mix a set of golden tasks in real tasks, then how much gain in truth inference can be benefited for each method? (Section 6.3.3)
4. What are the effects of different task types, task models, worker models, and inference techniques? In other words, what factors are beneficial to inferring the truth? (Section 6.3.4)

6.3.1 Varying Data Redundancy

For data redundancy, we define it as the number of answers collected for each task. In our 5 used datasets (Table 5), the data redundancy for each dataset is $|V|/n$. In Figures 4, 5, and 6, we observe the quality of each method in each dataset with varying data redundancy. For example, in Figure 4(a), on dataset `D_PosSent` (with $|V|/n = 3$), we compare with 14 methods that can be used in decision-making tasks (Table 4), i.e., MV, ZC, GLAD, D&S,

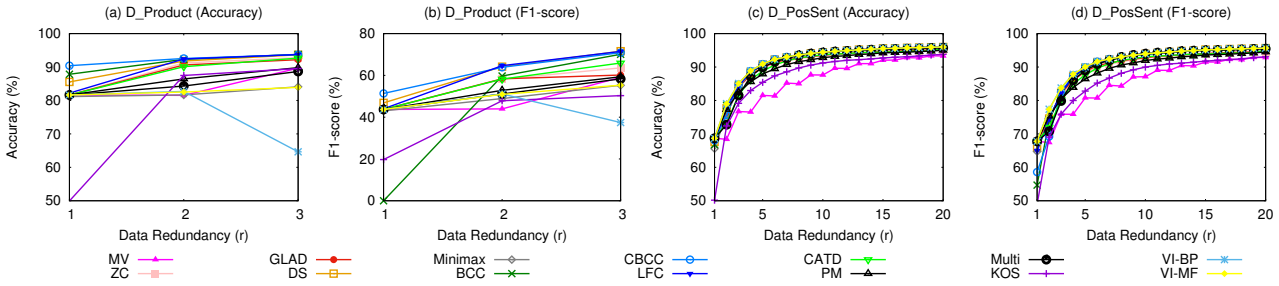

Figure 4: Quality Comparisons on Decision-Making Tasks (Section 6.3.1).

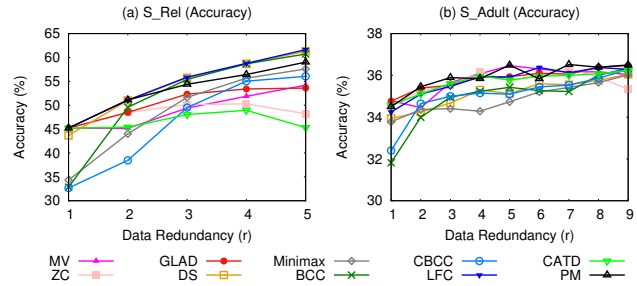

Figure 5: Quality Comparisons on Single-Label Tasks (Section 6.3.1).

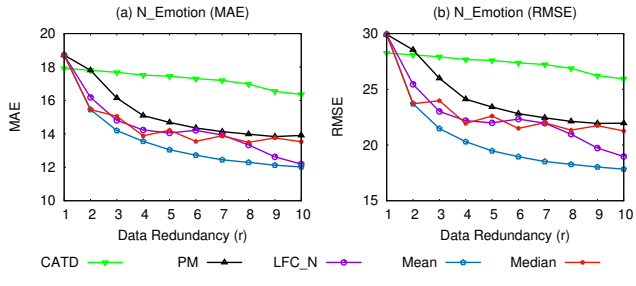

Figure 6: Quality Comparisons on Numeric Tasks (Section 6.3.1).

Table 6: The Quality and Running Time of Different Methods with Complete Data (Section 6.3.1).

Method	D_Product			D_PosSent			S_Rel		S_Adult		N_Emotion		
	Accuracy	*F1-score*	*Time*	*Accuracy*	*F1-score*	*Time*	*Accuracy*	*Time*	*Accuracy*	*Time*	*MAE*	*RMSE*	*Time*
MV	89.66%	59.05%	0.13s	93.31%	92.85%	0.08s	54.19%	0.49s	36.04%	0.40s	×	×	×
ZC [16]	92.80%	63.59%	1.04s	95.10%	94.60%	0.55s	48.21%	7.39s	35.34%	6.42s	×	×	×
GLAD [53]	92.20%	60.17%	907.11s	95.20%	94.71%	407.66s	53.59%	5850.39s	36.47%	4194.50s	×	×	×
D&S [15]	93.66%	**71.59%**	1.46s	**96.00%**	**95.66%**	0.80s	61.30%	10.67s	36.05%	9.18s	×	×	×
Minimax [61]	84.09%	55.26%	272.05s	95.80%	95.43%	35.71s	57.59%	1728.09s	36.03%	1223.75s	×	×	×
BCC [27]	**93.78%**	70.10%	9.82s	**96.00%**	**95.66%**	6.06s	60.72%	153.50s	36.34%	137.92s	×	×	×
CBCC [46]	93.72%	70.87%	5.53s	**96.00%**	**95.66%**	4.12s	56.05%	44.69s	36.28%	42.52s	×	×	×
LFC [41]	93.73%	71.48%	1.42s	**96.00%**	**95.66%**	0.83s	**61.64%**	10.75s	36.29%	9.26s	×	×	×
CATD [30]	92.66%	65.92%	2.97s	95.50%	95.07%	1.32s	45.32%	16.13s	36.23%	12.96s	16.36	25.94	2.15s
PM [5, 31]	89.81%	59.34%	0.56s	95.04%	94.53%	0.33s	59.02%	2.60s	**36.50%**	2.09s	13.91	21.96	0.36s
Multi [51]	88.67%	58.32%	15.48s	95.70%	95.44%	4.98s	×	×	×	×	×	×	×
KOS [26]	89.55%	50.31%	24.06s	93.80%	93.06%	10.14s	×	×	×	×	×	×	×
VI-BP [33]	64.64%	37.43%	306.23s	**96.00%**	**95.66%**	58.52s	×	×	×	×	×	×	×
VI-MF [33]	83.91%	55.31%	38.96s	**96.00%**	**95.66%**	6.71s	×	×	×	×	×	×	×
LFC_N [41]	×	×	×	×	×	×	×	×	×	×	12.20	18.97	0.23s
Mean	×	×	×	×	×	×	×	×	×	×	**12.02**	**17.84**	0.09s
Median	×	×	×	×	×	×	×	×	×	×	13.53	21.26	0.11s

Minimax, BCC, CBCC, LFC, CATD, PM, Multi, KOS, VI-BP and VI-MF. We vary the data redundancy $r \in [1, 3]$, where for each specific r, we randomly select r out of 3 answers collected for each task, and construct a dataset with the selected answers (i.e., a dataset with the number of answers $r \cdot n$ for all n tasks). Then we run each method on the constructed dataset and record the *Accuracy* based on comparing each method's inferred truth with the ground truth. We repeat each experiment for 30 times and the average quality is reported. As discussed in Section 6.1.2, we use metrics *Accuracy*, *F1-score* on decision-making tasks (D_Product, D_PosSent), metric *Accuracy* on single-label tasks (S_Rel, S_Adult) and metrics *MAE*, *RMSE* on numeric tasks (N_Emotion). To have a clear comparison, we record the quality and efficiency in the complete dataset (i.e., with redundancy $|V|/n$) for all methods in Table 6. Based on the results in Figures 4-6, and Table 6, we analyze the quality and efficiency of different methods.

(1) The Quality of Different Methods in Different Datasets.

Decision-Making Tasks. For dataset D_Product, i.e., Figures 4(a), (b), we can observe that (1) as the data redundancy r is varied in $[1, 3]$, the quality increases with r for different methods. (2) In Table 6, it can be observed that for *Accuracy*, the quality does not make significant differences between methods (most methods' quality are around 90%); while for *F1-score*, it makes differences, and only 4 methods' quality (D&S, BCC, CBCC, LFC) are above 70%, leading more than 4% compared with other methods. We have analyzed in Section 6.1.2 that *F1-score* is more meaningful to D_Product compared with *Accuracy*, as we are more interested in finding out the "*same*" products. (3) In terms of task models, incorporating task difficulty (GLAD) or latent topics (Minimax) do not bring significant benefits in quality. (4) In terms of worker models, we can observe that the four methods with confusion matrices (i.e., D&S, BCC, CBCC, LFC) perform significantly better than other methods with worker probability. The reason is that confusion matrix models each worker as a 2×2 matrix q^w in decision-making tasks, which captures both $q^w_{1,1} = \Pr(v^w_i = \mathrm{T} \mid v^*_i = \mathrm{T})$, i.e., the probability that a worker w answers correctly if the truth is T and $q^w_{2,2} = \Pr(v^w_i = \mathrm{F} \mid v^*_i = \mathrm{F})$, i.e., the probability that w answers correctly if the truth is F. However, the worker probability models a worker as a single value, which substantially assumes that $q^w_{1,1} = q^w_{2,2}$ in confusion matrix. This cannot fully capture a worker's answering performance. Note that in D_Product, typically workers have high values for $q^w_{2,2}$ and low values for $q^w_{1,1}$. Since for a pair of different products, if one difference is spotted between them, then it will be answered correctly, which is easy ($q^w_{2,2}$ is high); while for a pair of same products, it will be answered correctly only if all the features in the products are spotted the same,

which is hard ($q^w_{1,1}$ is low). Although VI-BP and VI-MF also use confusion matrix, they perform bad, probably because they leverage *Variational Inference* to infer the parameters, which may derive workers' qualities wrongly. For other worker models such as worker bias (Multi), worker variance (Multi, LFC_N), diverse skills (Multi, Minimax) and confidence (CATD), they do not outperform the confusion matrix methods in quality, probably due to the fact that the methods cannot infer those parameters correctly. (5) For BCC, the *F1-score* is 0 as $r = 1$, since BCC returns all tasks as F. which gives no information to T. However, in Table 6 (with completed data), the method BCC performs the best in *Accuracy*, while the method D&S performs the best in *F1-score*.

For dataset D_PosSent, i.e., Figures 4(c),(d), it can be observed that (1) as r is varied in $[1, 10]$, the quality increases significantly with r for different methods (improving around 20%), and then have a minor increase ever since ($r \in [11, 20]$). (2) Similar to D_Product, the six methods with confusion matrix as worker models (i.e., D&S, BCC, CBCC, LFC, VI-BP and VI-MF) perform equally the best, since confusion matrix captures more information than worker probability. However, other methods with more complicated task models and worker models do not express their benefits in quality. (3) *Accuracy* and *F1-score* in D_PosSent do not have significant differences, since unlike D_Product, the #tasks with T and F as truth in D_PosSent (i.e., 528 and 472) is balanced.

Single-Label Tasks. In Figure 5, on datasets S_Rel and S_Adult, we compare with 10 methods that specifically address single-label tasks (Table 4), i.e., MV, ZC, GLAD, D&S, Minimax, BCC, CBCC, LFC, CATD, and PM. We have the following observations: (1) on S_Rel, in general, the quality of methods increase with r; while on S_Adult, the quality of methods increase with $r \in [1, 5]$, and keep stable for $r \geq 5$. (2) In terms of quality, on S_Rel, the three methods D&S, BCC and LFC with quality $\geq 60\%$ outperform the other methods; while on S_Adult, the performance of different methods are similar. (3) On S_Rel, the quality of methods CATD and ZC decrease as $r \geq 4$, probably because they are sensitive to low quality workers' answers. (4) The quality of methods for single-label tasks are lower than that for decision-making tasks, since workers are not good at answering tasks with multiple choices, and the methods for single-label tasks are sensitive to low quality workers.

Numeric Tasks. In Figure 6, we compare with 5 methods that specifically address numeric tasks (Table 4): CATD, PM, LFC_N, Mean and Median. Note that *MAE* and *RMSE* are defined as errors, and the lower, the better. We have the following observations: (1) generally the errors of almost all methods decrease with the increasing r. (2) Among all methods, the baseline method Mean performs best, which regards each worker as equal. This means that workers' qualities may not be accurately inferred in CATD, PM and LFC_N. (3) It can be seen that the methods for numeric tasks are not well-addressed, as only 3 methods (i.e., CATD, PM, LFC_N) are specifically devised for numeric tasks.

(2) The Efficiency of Different Methods.

In terms of efficiency, some methods (MV, Mean and Median) can infer the truth directly, while other existing works (ZC, GLAD, D&S, Minimax, BCC, CBCC, LFC, CATD, PM, Multi, KOS, VI-BP, VI-MF, LFC_N) follow the iterative framework (Algorithm 1) until convergence is attained. For the iterative methods, the time complexity can be expressed as $\mathcal{O}(c \cdot t)$, where c is the #iterations to converge, and t is the time in each iteration. Thus a method is inefficient if it takes many iterations to converge, or the time is slow in each iteration. We record each method's efficiency in Table 6. We can observe that non-iterative methods (MV, Mean and Median) are finished within 1s. For iterative methods, (1) ZC, D&S, LFC,CATD, PM, LFC_N can finish within 15s, which is efficient. The reason is that they have a small c and t. (2) The methods BCC, CBCC, Multi, KOS and VI-MF take more than 15s, but less than 3min to finish the process. The reason is that for BCC, CBCC and VI-MF, they take many iterations c to converge; while Multi and KOS take a long time t in each iteration. (3) There are methods GLAD, Minimax and VI-BP that take up to 100min to finish, which is slow. The reason is that they solve an optimization function in each iteration, e.g., GLAD uses gradient descent [28] to update parameters in each iteration, which takes much time t.

Summary. We summarize based on the above results. (1) The quality increases significantly with small data redundancy r, and keeps stable after a certain redundancy $\widehat{r} > r$. Note that $\widehat{r}$ varies in different methods on different datasets. (2) There is no method that performs consistently the best on all tested datasets. (3) In decision-making and single-label tasks, in general the three methods (D&S, BCC, LFC) perform better than others with complete data (Table 6). Note D&S [15] is the most classical approach proposed in 1979, and all other methods (BCC and LFC) extend D&S in different perspectives. (4) In numeric tasks, they are not well-addressed in existing works, where the baseline method Mean performs best in N_Emotion. (5) In terms of task models, the methods that model task difficulty (GLAD) or latent topics (Multi) in tasks do not perform significantly better in quality; moreover, they often take more time to converge. (6) In terms of worker models, generally speaking, confusion matrix performs better in quality compared with worker probability; while other worker models (e.g., diverse skills, worker bias, variance and confidence) do not bring significant benefits. Not surprisingly, methods with complicated worker models often lead to inefficiency. (7) In terms of inference techniques, for effectiveness, the methods with Optimization and PGM are more effective than Direct Computation. For efficiency, Direct Computation is more efficient than Optimization and PGM.

6.3.2 The Effect of Qualification Test

We next study how each method can be affected by qualification test. In real crowdsourcing platforms (e.g., AMT [2]), a fixed set of golden tasks can be set for each worker to answer when the worker first comes to answer tasks. We collect dataset D_PosSent from AMT [2], where each worker is required to answer 20 tasks with known ground truth (qualification test) when she first comes. However, in the other four public datasets, the data for qualification test are not made public (or not used in most cases). Thus, (1) we first *simulate* each worker's answers for qualification test; (2) then use each worker's answering performance for them to initialize the worker's quality (line 1 in Algorithm 1); (3) finally we run each method with the initialized worker's quality. For example, in D_Product, for each worker w, in her answers for all tasks ($\mathcal{T}^w$), we use bootstrap sampling [18], i.e., *sample with replacement* to sample 20 times, where each time we randomly sample her answer for one task (as there may be limited answers for a worker, thus bootstrap sampling is used, which can uncover the real distribution, i.e., worker's quality). Then we assume the 20 tasks' truth are known, and use her answering performance to initialize her quality.

To leverage a worker's answers for qualification test, for example, in ZC [16], as each worker is modeled as worker probability, then her quality is initialized as the fraction of correctly answered tasks in these 20 sampled ones. Finally the two steps in ZC are iteratively run until convergence, and the quality w.r.t. ground truth is computed. We find that there are only 8 methods (i.e., ZC, GLAD, D&S, LFC, CATD, PM, VI-MF and LFC_N) that can initialize workers' qualities using qualification test. For these methods, we repeat each experiment for 100 times. We denote $\widetilde{c}$ as the average quality with qualification test; c as the quality without qualification test (i.e., in Table 6); and $\Delta = \widetilde{c} - c$ as the improvement of qual-

Table 7: The Quality with Qualification Test ($\widetilde{c}$) and Benefit ($\Delta = \widetilde{c} - c$) of Different Methods in Each Dataset (Section 6.3.2).

Method	D_Product (Simulation)		D_PosSent (Real)		S_Rel (Simulation)	S_Adult (Simulation)	N_Emotion (Simulation)	
	Accuracy (Δ)	*F1-score* (Δ)	*Accuracy* (Δ)	*F1-score* (Δ)	*Accuracy* (Δ)	*Accuracy* (Δ)	*MAE* (Δ)	*RMSE* (Δ)
ZC	92.95% (+0.15%)	64.4% (+0.81%)	95.10% (0.00%)	94.60% (0.00%)	55.24% (+7.03%)	35.54% (+0.20%)	×	×
GLAD	92.18% (–0.02%)	60.04% (–0.03%)	95.20% (0.00%)	94.71% (0.00%)	53.48% (–0.19%)	36.30% (+0.43%)	×	×
D&S	93.98% (+0.32%)	72.43% (+0.84%)	95.90% (–0.10%)	95.55% (–0.11%)	61.42% (+0.12%)	36.93% (+1.16%)	×	×
LFC	93.98% (+0.25%)	72.43% (+0.95%)	95.90% (–0.10%)	95.55% (–0.11%)	61.42% (+0.12%)	36.93% (+1.16%)	×	×
CATD	93.11% (+0.45%)	67.48% (+1.56%)	95.50% (+0.01%)	95.07% (+0.01%)	44.09% (–1.26%)	35.68% (–0.50%)	17.97 (+1.61)	28.56 (+2.62)
PM	90.55% (+0.74%)	61.26% (+1.92%)	95.10% (+0.05%)	94.60% (+0.06%)	59.41% (+0.39%)	36.70% (+0.41%)	17.27 (+3.36)	27.42 (+5.46)
VI-MF	85.26% (+1.35%)	57.31% (+2.00%)	95.90% (–0.10%)	95.54% (–0.12%)	×	×	×	×
LFC_N	×	×	×	×	×	×	12.20 (0.00)	18.97 (0.00)

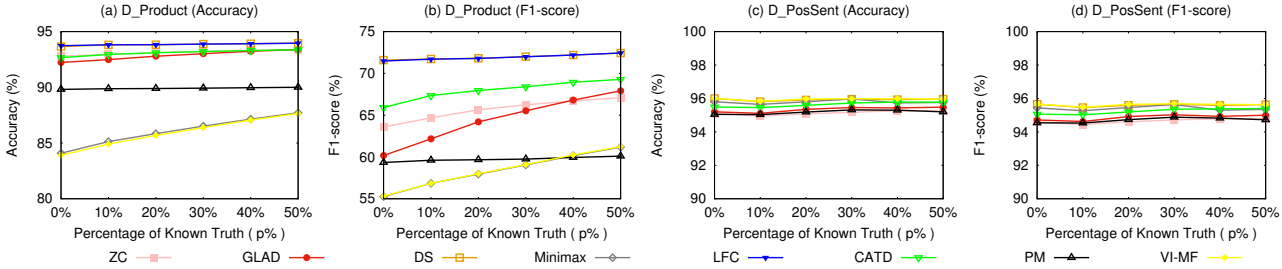

Figure 7: Varying Hidden Test on Decision-Making Tasks (Section 6.3.3).

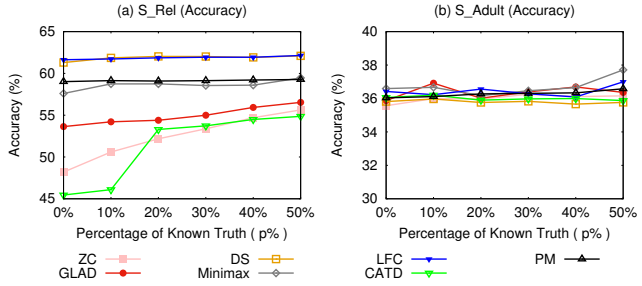

Figure 8: Varying Hidden Test on Single-Label Tasks (Section 6.3.3).

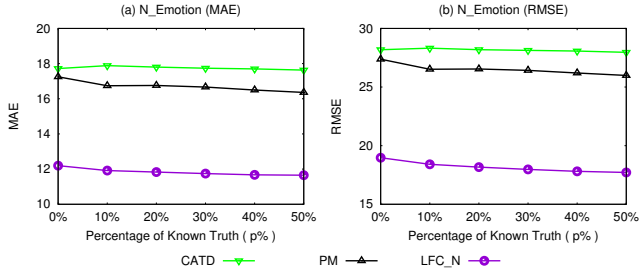

Figure 9: Varying Hidden Test on Numeric Tasks (Section 6.3.3).

ity. Table 7 shows $\widetilde{c}$ and Δ of each method in each dataset. Note that the only difference between c and $\widetilde{c}$ is that they use different initializations of workers' qualities (line 1 in Algorithm 1).

Decision-Making & Single-Label Tasks. It can be observed that no matter in real qualification test (D_PosSent), or simulations (D_Product, S_Rel, S_Adult), not all methods can benefit from qualification test and the benefits vary in different datasets. For example, in D_Product and S_Adult, almost all methods can benefit; while in D_PosSent, only PM and CATD can benefit. The reason is that in D_Product, each task is only answered by 3 workers, while in D_PosSent, each task is answered by 20 workers. The dataset with small data redundancy (e.g., D_Product) requires qualification test for a good initialization of workers' qualities, while other datasets can correctly detect each worker's quality in an unsupervised manner. We can also observe that the benefit (Δ) is often small and sometimes $\Delta < 0$, since almost all methods adopt an iterative approach and approximate the objective value, thus an inadequate initialization may lead to a bad local optimum.

Numeric Tasks. For dataset N_Emotion, even no methods in CATD, PM and LFC_N can benefit, where both the errors *MAE* and *RMSE* increase for all methods. As mentioned before, this is probably because the methods are not studied properly in the numeric data, and the qualities modeled for workers are not accurate enough.

Summary. (1) Some methods can benefit from the qualification test with marginal benefit. (2) In numeric tasks, most of methods cannot benefit, and there is still room for improvement. (3) There are some methods that are hard to incorporate qualification test.

6.3.3 *The Effect of Hidden Test*

We evaluate how hidden test affects each method's quality. Given V, suppose we also know the set of golden tasks $\mathcal{T}' \subseteq \mathcal{T}$, then how much quality can be improved using existing methods? To implement this idea, we take a look at existing methods (Table 4) and observe that there are 9 methods (ZC, GLAD, D&S, Minimax, LFC, CATD, PM, VI-MF, and LFC_N) that can be easily extended to incorporate the golden tasks into its iterative algorithm.

To incorporate the golden tasks, consider Algorithm 1, in step 1, we only update the truth of tasks with unknown truth; in step 2, we update each worker's quality by considering both the truth of golden tasks and other tasks' inferred truth in step 1. We show the quality of different methods by varying the size of golden tasks ($\mathcal{T}'$) in Figures 7, 8 and 9. For example, in Figure 7(a), on dataset D_Product, we randomly select $p\%$ in the task set $\mathcal{T}$ as the golden tasks ($\mathcal{T}'$). Then we take $\mathcal{T}'$ and workers' answers V as the input to different methods, and further test different methods' quality by comparing the inferred truth of $\mathcal{T} - \mathcal{T}'$ with their ground truth. We vary $p \in [0, 50]$, where for each p, we repeat each experiment 100 times and record the average quality. Next, we analyze the results.

Decision-Making & Single-Label Tasks. In Figures 7 and 8, it can be seen that (1) generally starting from $p = 0$ (Table 6), the quality of methods increase with p, since knowing more truth is beneficial to more accurate estimation. (2) The methods on dataset D_PosSent do not have significant gains with varying p, since each task is answered by multiple workers, and the inferred parameters are hard to be affected by the known truth. (3) Only a few methods (e.g., ZC, CATD) are sensitive to golden tasks, since most iterative methods are easy to fall into local optimum.

Numeric Tasks. In Figure 9, we compare with three methods (LFC_N, CATD, PM) in N_Emotion and we find that the errors (*MAE* and *RMSE*) decrease slightly with the increasing p.

Summary. (1) Generally the quality of different methods increase with more proportion ($p\%$) of golden tasks. (2) Different methods have different improvements on different datasets. (3) Only 9 methods are easy to incorporate golden tasks.

6.3.4 Analyzing Different Factors in Truth Inference

Task Types. In terms of task types, we observe that the methods for decision-making tasks have been studied a lot and already have very good performance. Compared with decision-making tasks, the methods for single-label tasks do not perform well, e.g., the qualities for `S_Rel` and `S_Adult` are much lower than those for `D_Product` and `D_PosSent` in Table 6, since the methods for single-label tasks are more sensitive to workers with low qualities. For numeric tasks, the methods are not studied very well, and even the baseline method Mean outperforms others in dataset `N_Emotion`. This is because that on one hand, few methods specifically study numeric tasks; on the other hand, most methods cannot estimate workers' qualities for numeric tasks accurately.

Task Models. In terms of task models, GLAD and Minimax are the only two methods (Table 4) that consider specific task models (task difficulty and latent topics, respectively). However, they do not show improvements in quality compared with other methods with no task models. Moreover, they often take a long time to converge (e.g., $>$ 1000s in `S_Rel` and `S_Adult`). This is probably due to that their inference methods are not robust, and in some cases they cannot estimate the parameters in task models accurately. The incorporation of task models also leads to inefficiency.

Worker Models. In terms of worker models, in general, methods with confusion matrix (D&S, BCC, CBCC, LFC, VI-BP, VI-MF) perform better than methods with worker probability (ZC, GLAD, CATD, PM, KOS), since confusion matrix is more expressive than worker probability. Note that the quality of methods also vary a lot even if they apply the same worker model, e.g., for confusion matrix, the methods D&S, BCC, LFC are more robust than others (CBCC, VI-BP, VI-MF), since their techniques can infer workers' qualities more accurately. For other worker models, e.g., worker bias (Multi), worker variance (Multi, LFC_N), diverse skills (Multi, Minimax) and confidence (CATD), they do not achieve higher gains in quality. We also observe that not necessarily "*the more complex the model is, the higher quality the method will achieve*". For example, although Multi considers diverse skills, worker bias and variance in its worker models, the quality does not bring significant benefits. Ideally more complicated worker models lead to much higher quality; however, this introduces more computational complexity, and on one hand, it is challenging to estimate a large set of parameters accurately; on the other hand, it is hard to converge. Thus most methods with complicated worker models fail to achieve very good performance in quality and efficiency.

Inference Techniques. We analyze the techniques from quality, efficiency and interpretability, respectively. (1) In terms of quality, the methods with Optimization and PGM are more effective than the methods with Direct Computation, as they consider more parameters and study how to infer them iteratively. (2) In terms of efficiency, methods with Optimization and PGM are less efficient than methods with Direct Computation. Different optimization functions often vary significantly in efficiency, e.g., *Bayesian Estimator* is less efficient than *Point Estimation*, and some techniques (e.g., *Gibbs Sampling*, *Variational Inference*) often take a long time to converge. (3) In terms of interpretability, Optimization is easier to understand. The reason is that people can interpret the relations between worker's quality and task's truth in the self-defined optimization function. For PGM, it should conform to the model (Figure 1), which gives less freedom to express the optimization function. Moreover, it is hard to devise an easily solvable optimization function, and the developed iterative algorithms often lead to local optimum (e.g., Expectation Maximization [15, 17]).

7. CONCLUSION & FUTURE DIRECTIONS

We provide a detailed survey on truth inference in crowdsourcing and perform an in-depth analysis of 17 existing methods. We summarize a framework (Algorithm 1) and analyze the *task types*, *task models*, *worker models* and *inference techniques* in these methods. We also conduct sufficient experiments to compare these methods on 5 datasets with varying task types and sizes. Based on the analysis and experimental results, we have the following suggestions.

Decision-Making & Single-Label Tasks. If one has sufficient workers' answers (e.g., with redundancy over 20), and wants a very simple implementation that attains reasonable results, then we recommend the baseline method, i.e., MV; if one wants an implementation with little overhead but attains very good results, then we recommend the classical method D&S [15], which is robust in practice; if one would like to try some extensions of D&S, then BCC [27] and LFC [41] are good choices; if one wants to learn more inference techniques and incorporate various task/worker models, we recommend the PGM method Multi [51] and the Optimization method Minimax [61].

Numeric Tasks. If one has sufficient workers' answers, we recommend the baseline method (i.e., Mean); if one wants to learn more advanced techniques and worker models, we recommend the PGM method LFC_N [41], and the Optimization method CATD [30].

We also point out the following future research directions.

(1) Task Types. In decision-making and single-label tasks, there is no "best" method that beats others, and we recommend D&S [15] and LFC [41], which are relatively more effective and efficient. In numeric tasks, we recommend Mean and LFC_N [41], and there is still room to improve numeric tasks. Moreover, there are other more complicated task types that are merely studied, e.g., translation tasks [63], or tasks that require workers to collect data [48].

(2) Task Design. In order to collect high quality crowdsourced data in an efficient way, it is important to design tasks with friendly User Interface (UI) with a feasible price. Although there are some works that study how to set the prices [21] and acquire answers from crowd more efficiently [23], the design of friendly UI is not studied extensively. It is also interesting to study the relations between the design of UI, price, worker's latency and quality.

(3) Data Redundancy. The quality significantly increases with small redundancy, and keeps stable for a large redundancy. Then how to estimate the data redundancy with stable quality? Is it possible to estimate the improvement with more data redundancy?

(4) Qualification Test. Not all methods can benefit from qualification test, and the quality of some methods even decrease. So is it possible to estimate the benefit of qualification test for a method?

(5) Hidden Test. Although most methods can benefit from them, the improvements vary in different datasets and methods. Thus is it possible to estimate the improvement with hidden test (i.e., a number of golden tasks) for a method on a dataset?

(6) Task Assignment. In this paper, we focus on a static problem which takes workers' answers as input. Recently, there are some works [60, 19, 34, 7] that study how to assign tasks to appropriate workers in an online manner, called *Online Task Assignment*. It is interesting to see how the answers collected by different task assignment strategies can affect the truth inference quality.

(7) Incorporation of More Rich Features. In this paper, we only consider the collected answers for tasks; however, we do not incorporate more rich information, for example, the contexts in tasks (each task's textual descriptions or the pixels in image tasks). These have been mentioned in existing works [19, 35, 54, 43]. We do not include those into comparisons since most of the datasets do not

make the original tasks public. It might be interesting to see how much improvement when such information is considered.

(8) Benchmark. In database research, there has been various benchmarks (e.g., TPC-C for OLTP performance), which provide a standardized measure to evaluate the performance of different methods. However, in crowdsourcing area, although there has been some public datasets [13], few benchmarks are available. It is important to develop benchmarks and evaluation measures in crowdsourcing.

Acknowledgement. Guoliang Li and Yuanbing Li were supported by 973 Program of China (2015CB358700), NSF of China (61373024, 61632016, 61422205, 61472198), Shenzhou, Tencent, FDCT/116/2013/A3, and MYRG105 (Y1-L3)-FST13-GZ. Reynold Cheng, Yudian Zheng, and Caihua Shan were supported by the Research Grants Council of Hong Kong (RGC Projects HKU 17229116 and 17205115) and University of Hong Kong (Projects 102009508 and 104004129).

8. REFERENCES

[1] https://docs.aws.amazon.com/AWSMechTurk/latest/RequesterUI/amt-ui.pdf.
[2] Amazon mechanical turk. https://www.mturk.com/.
[3] Chi-squared. https://en.wikipedia.org/wiki/Chi-squared_distribution.
[4] Adult Datset. https://github.com/ipeirotis/Get-Another-Label/tree/master/data.
[5] B. I. Aydin, Y. S. Yilmaz, Y. Li, Q. Li, J. Gao, and M. Demirbas. Crowdsourcing for multiple-choice question answering. In *AAAI*, pages 2946–2953, 2014.
[6] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *JMLR*, 3(Jan):993–1022, 2003.
[7] R. Boim, O. Greenshpan, T. Milo, S. Novgorodov, N. Polyzotis, and W.-C. Tan. Asking the right questions in crowd data sourcing. In *ICDE*, pages 1261–1264, 2012.
[8] D. G. Brizan and A. U. Tansel. A. survey of entity resolution and record linkage methodologies. *Communications of the IIMA*, 6(3):5, 2015.
[9] C. Buckley, M. Lease, and M. D. Smucker. Overview of the trec 2010 relevance feedback track (notebook). In *The Nineteenth TREC Notebook*, 2010.
[10] C. Callison-Burch. Fast, cheap, and creative: evaluating translation quality using amazon's mechanical turk. In *EMNLP*, pages 286–295, 2009.
[11] C. Chai, G. Li, J. Li, D. Deng, and J. Feng. Cost-effective crowdsourced entity resolution: A partial-order approach. In *SIGMOD*, pages 969–984, 2016.
[12] CrowdFlower. http://crowdflower.com/.
[13] Crowdsourcing Datasets. http://dbgroup.cs.tsinghua.edu.cn/ligl/crowddata/.
[14] S. B. Davidson, S. Khanna, T. Milo, and S. Roy. Using the crowd for top-k and group-by queries. In *ICDT*, pages 225–236, 2013.
[15] A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied statistics*, pages 20–28, 1979.
[16] G. Demartini, D. E. Difallah, and P. Cudré-Mauroux. Zencrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In *WWW*, pages 469–478, 2012.
[17] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.
[18] B. Efron and R. J. Tibshirani. *An introduction to the bootstrap*. 1994.
[19] J. Fan, G. Li, B. C. Ooi, K.-l. Tan, and J. Feng. icrowd: An adaptive crowdsourcing framework. In *SIGMOD*, pages 1015–1030, 2015.
[20] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin. Crowddb: answering queries with crowdsourcing. In *SIGMOD*, pages 61–72, 2011.
[21] Y. Gao and A. Parameswaran. Finish them!: Pricing algorithms for human computation. *PVLDB*, 7(14):1965–1976, 2014.
[22] S. Guo, A. G. Parameswaran, and H. Garcia-Molina. So who won?: dynamic max discovery with the crowd. In *SIGMOD*, pages 385–396, 2012.
[23] D. Haas, J. Wang, E. Wu, and M. J. Franklin. Clamshell: Speeding up crowds for low-latency data labeling. *PVLDB*, 9(4):372–383, 2015.
[24] H. Hu, Y. Zheng, Z. Bao, G. Li, J. Feng, and R. Cheng. Crowdsourced poi labelling: Location-aware result inference and task assignment. In *ICDE*, pages 61–72, 2016.
[25] M. Joglekar, H. Garcia-Molina, and A. Parameswaran. Evaluating the crowd with confidence. In *SIGKDD*, pages 686–694, 2013.
[26] D. R. Karger, S. Oh, and D. Shah. Iterative learning for reliable crowdsourcing systems. In *NIPS*, pages 1953–1961, 2011.
[27] H.-C. Kim and Z. Ghahramani. Bayesian classifier combination. In *AISTATS*, pages 619–627, 2012.
[28] D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT Press, 2009.
[29] G. Li, J. Wang, Y. Zheng, and M. J. Franklin. Crowdsourced data management: A survey. *TKDE*, 28(9):2296–2319, 2016.
[30] 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.
[31] 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.
[32] B. Liu. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167, 2012.
[33] Q. Liu, J. Peng, and A. T. Ihler. Variational inference for crowdsourcing. In *NIPS*, pages 692–700, 2012.
[34] X. Liu, M. Lu, B. C. Ooi, Y. Shen, S. Wu, and M. Zhang. Cdas: a crowdsourcing data analytics system. *PVLDB*, 5(10):1040–1051, 2012.
[35] F. Ma, Y. Li, Q. Li, M. Qiu, J. Gao, S. Zhi, L. Su, B. Zhao, H. Ji, and J. Han. Faitcrowd: Fine grained truth discovery for crowdsourced data aggregation. In *KDD*, pages 745–754, 2015.
[36] A. Marcus, E. Wu, D. R. Karger, S. Madden, and R. C. Miller. Human-powered sorts and joins. *PVLDB*, 5(1):13–24, 2011.
[37] A. Marcus, E. Wu, S. Madden, and R. C. Miller. Crowdsourced databases: Query processing with people. In *CIDR*, pages 211–214, 2011.
[38] A. G. Parameswaran, H. Garcia-Molina, H. Park, N. Polyzotis, A. Ramesh, and J. Widom. Crowdscreen: Algorithms for filtering data with humans. In *SIGMOD*, pages 361–372, 2012.
[39] A. G. Parameswaran, H. Park, H. Garcia-Molina, N. Polyzotis, and J. Widom. Deco: declarative crowdsourcing. In *CIKM*, pages 1203–1212, 2012.
[40] Project Page. http://dbgroup.cs.tsinghua.edu.cn/ligl/crowd_truth_inference/.
[41] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy. Learning from crowds. *JMLR*, 11(Apr):1297–1322, 2010.
[42] C. E. Shannon. A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5(1):3–55, 2001.
[43] E. D. Simpson, M. Venanzi, S. Reece, P. Kohli, J. Guiver, S. J. Roberts, and N. R. Jennings. Language understanding in the wild: Combining crowdsourcing and machine learning. In *WWW*, pages 992–1002, 2015.
[44] R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *EMNLP*, pages 254–263, 2008.
[45] Twitter Sentiment. http://www.sananalytics.com/lab/twitter-sentiment/.
[46] M. Venanzi, J. Guiver, G. Kazai, P. Kohli, and M. Shokouhi. Community-based bayesian aggregation models for crowdsourcing. In *WWW*, pages 155–164, 2014.
[47] P. Venetis, H. Garcia-Molina, K. Huang, and N. Polyzotis. Max algorithms in crowdsourcing environments. In *WWW*, pages 989–998, 2012.
[48] L. Von Ahn, B. Maurer, C. McMillen, D. Abraham, and M. Blum. recaptcha: Human-based character recognition via web security measures. *Science*, 321(5895):1465–1468, 2008.
[49] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.
[50] J. Wang, T. Kraska, M. J. Franklin, and J. Feng. Crowder: Crowdsourcing entity resolution. *PVLDB*, 5(11):1483–1494, 2012.
[51] P. Welinder, S. Branson, P. Perona, and S. J. Belongie. The multidimensional wisdom of crowds. In *NIPS*, pages 2424–2432, 2010.
[52] S. E. Whang, P. Lofgren, and H. Garcia-Molina. Question selection for crowd entity resolution. *PVLDB*, 6(6):349–360, 2013.
[53] J. Whitehill, T.-f. Wu, J. Bergsma, J. R. Movellan, and P. L. Ruvolo. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *NIPS*, pages 2035–2043, 2009.
[54] Y. Yan, R. Rosales, G. Fung, M. W. Schmidt, G. H. Valadez, L. Bogoni, L. Moy, and J. G. Dy. Modeling annotator expertise: Learning when everybody knows a bit of something. In *AISTATS*, pages 932–939, 2010.
[55] X. Zhang, G. Li, and J. Feng. Crowdsourced top-k algorithms: An experimental evaluation. *PVLDB*, 9(8):612–623, 2016.
[56] W. X. Zhao, J. Jiang, J. Weng, J. He, E.-P. Lim, H. Yan, and X. Li. Comparing twitter and traditional media using topic models. In *ECIR*, pages 338–349. 2011.
[57] Z. Zhao, F. Wei, M. Zhou, W. Chen, and W. Ng. Crowd-selection query processing in crowdsourcing databases: A task-driven approach. *EDBT*, pages 397–408, 2015.
[58] Y. Zheng, R. Cheng, S. Maniu, and L. Mo. On optimality of jury selection in crowdsourcing. In *EDBT*, pages 193–204, 2015.
[59] Y. Zheng, G. Li, and R. Cheng. Docs: a domain-aware crowdsourcing system using knowledge bases. *PVLDB*, 10(4):361–372, 2016.
[60] Y. Zheng, J. Wang, G. Li, R. Cheng, and J. Feng. Qasca: A quality-aware task assignment system for crowdsourcing applications. In *SIGMOD*, pages 1031–1046, 2015.
[61] D. Zhou, S. Basu, Y. Mao, and J. C. Platt. Learning from the wisdom of crowds by minimax entropy. In *NIPS*, pages 2195–2203, 2012.
[62] D. Zhou, Q. Liu, J. Platt, and C. Meek. Aggregating ordinal labels from crowds by minimax conditional entropy. In *ICML*, pages 262–270, 2014.
[63] S. Zhu, Y. Wu, and D. Mumford. Minimax entropy principle and its application to texture modeling. *Neural computation*, 9(8):1627–1660, 1997.