CrowdSelect: Increasing Accuracy of Crowdsourcing Tasks through Behavior Prediction and User Selection

Chenxi Qiu
College of Information Sciences and Technology
Pennsylvania State University
University Park, PA
czq3@psu.edu

Anna C. Squicciarini
College of Information Sciences and Technology
Pennsylvania State University
University Park, PA
acs20@psu.edu

Barbara Carminati
Department of Theoretical and Applied Sciences
Universiy of Insubria
Varese, Italy
barbara.carminati@uninsubria.it

James Caverlee
Department of Computer Science and Engineering
Texas A&M University
College Station, TX
caverlee@cse.tamu.edu

Dev Rishi Khare
College of Information Sciences and Technology
Pennsylvania State University
University Park, PA
dzk5384@psu.edu

ABSTRACT

Crowdsourcing allows many people to complete tasks of various difficulty with minimal recruitment and administration costs. However, the lack of participant accountability may entice people to complete as many tasks as possible without fully engaging in them, jeopardizing the quality of responses. In this paper, we present a dynamic and time efficient solution to the task assignment problem in crowdsourcing platforms. Our proposed approach, CrowdSelect, offers a theoretically proven algorithm to assign workers to tasks in a cost efficient manner, while ensuring high accuracy of the overall task. In contrast to existing works, our approach makes minimal assumptions on the probability of error for workers, and completely removes the assumptions that such probability is known apriori and that it remains consistent over time. Through experiments over real Amazon Mechanical Turk traces and synthetic data, we find that CrowdSelect has a significant gain in term of accuracy compared to state-of-the-art algorithms, and can provide a 17.5% gain in answers' accuracy compared to previous methods, even when there are over 50% malicious workers.

1. INTRODUCTION

Crowdsourcing marketplaces have successfully leveraged the attention of millions of users to tackle traditionally repetitive problems that are difficult to automate. From specialized systems like Ushahidi (for crisis mapping), Foldit (for protein folding) and Duolingo (for translation) to general-purpose crowdsourcing platforms like Amazon Mechanical Turk and Crowdflower , crowdsourcing has emerged as an effective new model of distributed computing.

CIKM'16 , October 24-28, 2016, Indianapolis, IN, USA
© 2016 ACM. ISBN 978-1-4503-4073-1/16/10...$15.00
DOI: http://dx.doi.org/10.1145/2983323.2983830

The success of any given crowdsourced task is based on a large number of diverse workers, whose profiles vary considerably across several dimensions (e.g., skills, motivations, socio-economic backgrounds). These variations, along with the presence of dishonest workers, may likewise result in variable quality of obtained responses [13, 14, 29]. Accordingly, quality control for crowdsourced work is a primary concern [22]

The problem of quality control in crowdsourcing has been investigated both by crowdsourcing platform providers as well as by the research community. On one hand, crowdsourcing platforms now offer methods to filter workers' responses through quality checks [9, 25, 28, 34]. For instance, workers may be preselected based on their reputation or based on responses to a pre-task questionnaire assigned by the task requesters to gather data about workers' skills or biases [34]. On the other hand, researchers have recently proposed several approaches based on sophisticated statistical quality control methods, accounting for erroneous or lazy workers [7, 15, 17–19, 33, 38].

In this line, one approach to deal with quality control is to provide a strategic way to assign tasks to workers, in a manner such that quality of tasks is preserved [22]. In general, the problem of task-assignment involves selecting a set of workers among a set $\mathcal{U}$ of available workers for a given task t, of a specific category or domain [15, 17–19, 33]. Under the assumption that each worker i receives a compensation c_i^t for each completed assignment, this problem can be constrained by a budget B, namely the maximum budget that can be allocated to hire workers. In this scenario, solving the task assignment problem involves finding the best workers-to-task assignment, so as to maximize the accuracy of the answers and minimize the overall cost to the task requester. Note that workers may take up different behavioral models (i.e., honest, adversarial or simply erratic) and therefore provide answers of different quality. A worker's behavior with respect to expected quality of response is estimated as his probability of applying an incorrect label to the assigned task.

The goal of this paper is to investigate solutions for this problem in the presence of such a diverse set of workers, wherein the probability of error of any given worker is unknown. Estimating workers' probabilities of errors and selecting workers for given tasks are the two challenging parts of our solutions. In contrast to existing works [7, 15, 17–19, 33, 38], our approach, referred to as *CrowdSelect*, makes minimal assumptions on the probability of error for workers, and completely removes the assumptions that such probability is known apriori and that it remains consistent over time.

Our proposed assignment strategy takes into account individual differences, and is able to estimate on-the-fly how to leverage workers' answers, regardless of the quality of their work and their possible biases. In this way, even malicious workers' responses can be used to solve tasks. Tasks are assigned to workers according to the workers' estimated error rate such as to minimize estimation error of answers, and to meet budget constraints. As shown in our theoretical analysis, CrowdSelect is both effective and efficient and can provide task requesters with strong guarantees on the accuracy of their tasks, under different fusion methods, like majority voting and Bayesian classifier [3, 8, 20].

With respect to performance, results based on an Amazon Mechanical Turk experiment, i.e., over 6,000 tweets traces, demonstrate that CrowdSelect's accuracy outperforms (i.e., has at least 8.6% gain) that of state-of-the-art algorithms, e.g., [7, 15, 18, 20]. Furthermore, from a set of experiments carried out with synthetic data, CrowdSelect has at least 17.5% gain in answers' accuracy compared to the previous methods, when there are over 50% malicious workers. We achieve near 100% accuracy when most workers' answers are informative (i.e., their error rate is either close to 1 or 0), regardless of workers' actual intentions.

The remainder of the paper is organized as follows. The next Section presents our problem statement. In Section 3, we introduce the worker model and our approach to predict workers' error rates. In Section 4, we present the worker selection problem and solve it by introducing our CrowdSelect algorithm. In Section 5, we evaluate the performance of CrowdSelect using both a real dataset and synthetic data, with comparisons to existing approaches. Finally, we present related work in Section 6 and conclude in Section 7.

2. OVERALL APPROACH

We propose a dynamic assignment model that accounts for workers behavior, as they complete tasks and take upon a certain behavioral pattern, be it honest, adversarial, or erratic. Here, and in the remaining of the paper, a task is an individual binary question (i.e., the answer can be either 1 or -1), which workers are asked to answer.

Our approach, which we refer to as *CrowdSelect*, includes two main components: *worker error rate prediction* and *worker selection*. The goal of the first component is to anticipate workers' error rates in a timely fashion, considering workers' behavioral changes over time. The second part, i.e., worker selection, aims at solving the task assignment problem by optimally selecting workers according to their predicted error rate.

The *worker error rate prediction* provides an estimate of a worker's expected behavior, according to his responses compared to the answers obtained for the same task from other workers and the answers of gold tasks (i.e., tasks for which the true answers is known to the task requester). Precisely, based on a worker's historical record, we predict his error rate for his upcoming tasks, through an auto-regressive

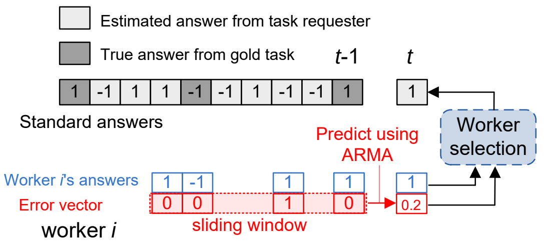


Figure 1: Error rate prediction framework ("-1, 1" in standard answers and workers' answers are the possible answer values, and "0, 1" in error vector represents whether workers answer correctly)

moving average (ARMA) model [6]. In addition, considering that solely using estimated answers to evaluate a worker's error rate may cause the evaluated error rate to deviate from the actual truth value, we also periodically assign a gold task test for each worker. Note that we do not require any prior assumption on the distribution of true labels for the tasks, but we assume, in line with previous work [33], that workers are independent and not colluding.

With respect to *worker selection*, as mentioned, we aim to select the optimal combination of workers, considering their cost and error rate, within some budget constraints. Solving the stated workers' selection problem is however computationally intractable. Accordingly, we confine the problem's objective function with three upper bound functions that have simpler forms (in Theorem 4.1), and relax the goal as minimizing the upper bound functions. To solve the relaxed problem, we propose a time-efficient algorithm (with time complexity $O(N \log N)$), which always chooses workers with informative answers, i.e., the error rate is close to either 0 or 1, and low cost. Finally, we derive the theoretical upper bounds of CrowdSelect's error rate (in Theorem 4.2 and Theorem 4.3).

3. WORKER MODEL AND ERROR RATE PREDICTION

Since Crowdsourcing is a human-powered problem solving paradigm, it is important to account for, and to the extent possible predict, the workers' quality of responses. Hence, our first challenge is to anticipate workers' error rate in a timely and accurate fashion.

Table 1: Notations

Notation	Description
T	The number of binary classification tasks
N	The number of workers
$\mathcal{U}$	The set of workers
p_i^t	The error rate of worker i for task t
Z_t	The true label of task t
$Y_{i,t}$	The answer of worker i to task t
$X_{i,t}$	Indicates whether worker i incorrectly answers task t If yes, $X_{i,t} = 1$; otherwise, $X_{i,t} = 0$
c_i^t	The cost of worker i for task t
B	The budget
L	The order of autoregressive terms in ARMA
D	The order of moving average terms in ARMA
w_i^t	The utility of worker i to task t

3.1 Estimating worker's errors rate through ARMA models

In this Section we present our overall framework for estimating workers' error rate as they complete tasks on a crowdsourcing platform.

3.1.1 *Overall framework and notations*

Our error prediction approach includes several steps, visually depicted in the framework of Figure 1. We maintain a *standard answer vector*, which stores the task requester's *estimated answers* (i.e., answers of tasks provided by workers only, which truth is unknown) as well as a small set of *gold task answers*. For each worker we also maintain an *error vector* to record whether the worker's answer is consistent with the standard answer from the past round. In particular, we consider a worker's answer as a "correct answer" (denoted by 0) if it is the same as the standard answer and an "incorrect answer" (denoted by 1) otherwise. Then, the error rate of each worker can be predicted according to the worker's error vector using an ARMA model.

In addition, considering that workers may have different capability of solving the tasks in different categories (e.g., workers without strong English skills may be uncomfortable in completing product reviews), we need to predict workers' error rate by category.

Table 1 lists the main notations that will be used throughout the model.

ARMA model. Let $1, ..., T$ denote the binary classification tasks assigned to the workers and the ground truth of task t is denoted by $Z_t \in \{-1, 1\}$ $(1 \leq t \leq T)$. Given N workers $\mathcal{U} = \{1, 2, ..., N\}$, worker i's answer for task t is denoted by $Y_{it} \in \{-1, 1\}$. We also represent the probability that worker i incorrectly answers task t by p_i^t, i.e., $p_i^t = \Pr\{Y_{it} \neq Z_t\}$, which is also called the *error rate* of worker i for task t.

To estimate p_i^t, we first implement a "gold task test" for each worker, where the correct answers in the test are known to the task requester. Workers' error rate may increase after the test, i.e., there is no continued incentive for workers to submit correct answers after they finish the test. Hence, we keep evaluating each worker's error rate after the gold task test and keep updating these values to reflect workers' most recent behavior.

For each new task, we consider only the worker's L most recent answers as a sample for prediction.[1] In order to determine L, we model each worker's trend of behavior by fitting an ARMA model [6]. $ARMA(L, D)$ refers to the model with L *autoregressive* terms $\{p_i^{t-L}, ..., p_i^{t-1}\}$ and D *moving-average* terms $\{\varepsilon_i^{t-D}, ..., \varepsilon_i^{t-1}\}$ ($\varepsilon_i^1, ..., \varepsilon_i^{t-1}$ is a noise process assumed to be uncorrelated with a zero mean and finite variance [6]), and $\{p_i^t\}$ is an ARMA(L, D) process if $\{p_i^t\}$ is stationary and satisfies the following relationship for every t

$$p_i^t - \phi_1 p_i^{t-1} - ... - \phi_L p_i^{t-L} = \varepsilon_i^t + \theta_1 \varepsilon_i^{t-1} + ... + \theta_D \varepsilon_i^{t-D} \quad (1)$$

where $\theta_1, ..., \theta_D, \phi_1, ..., \phi_L$ are the parameters of the model.

The data is assumed to be stationary in the ARMA model fitting procedure. If the time series exhibit variations that violate the stationary assumption, as is often the case with real workers' traces, then a "differencing operation" can be used:

$$\triangledown p_i^t = p_i^t - p_i^{t-1}. \quad (2)$$

If the non stationary part of a time series is a polynomial function of time, then differencing finitely many times can reduce the time series to an ARMA process [6].

Note that in practice, extremely erratic behavior cannot be captured by the ARMA model. As such, when estimating workers' error rate, we need to set upper and lower *error rate thresholds*, γ_{high} and γ_{low}, to remove workers with unstable behaviors. These thresholds should be set carefully, to ensure that only truly erratic behavior is disregarded. To determine the values of γ_{high} and γ_{low}, we first estimate each worker i's error rate $\hat{p}_i$ from pre-screen tasks, and then categorize workers into *unpredictable workers* $\mathcal{U}_{\text{unpr}}$ (including random workers and malicious workers with unpredictable behavior) and *predictable workers* $\mathcal{U}_{\text{pred}}$ (including reliable and malicious workers with predictable behavior) based on their estimated error rate. Next, we find the thresholds that can best separate these two types of workers, and use these values as the error rate thresholds for the remaining tasks. More specifically, we use $\mathcal{N}_{\text{pred}}^{\text{in}}(\gamma_{\text{low}}, \gamma_{\text{high}})$ to represent the number of predicted workers included by γ_{low} and γ_{high}, and $\mathcal{N}_{\text{pred}}^{\text{in}}(\gamma_{\text{low}}, \gamma_{\text{high}})$ is obtained by

$$\mathcal{N}_{\text{pred}}^{\text{in}}(\gamma_{\text{low}}, \gamma_{\text{high}}) = \sum_{i \in \mathcal{U}_{\text{pred}}} \left(\mathcal{H}(\hat{p}_i - \gamma_{\text{high}}) + \mathcal{H}(\gamma_{\text{low}} - \hat{p}_i)\right) \quad (3)$$

where $\mathcal{H}(\cdot)$ is step function, defined by

$$\mathcal{H}(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \quad (4)$$

Similarly, we use $\mathcal{N}_{\text{unpr}}^{\text{ex}}(\gamma_{\text{low}}, \gamma_{\text{high}})$ to represent the number of unpredicted workers excluded by γ_{low} and γ_{high}, and $\mathcal{N}_{\text{unpr}}^{\text{ex}}(\gamma_{\text{low}}, \gamma_{\text{high}})$ is obtained by

$$\mathcal{N}_{\text{unpr}}^{\text{ex}}(\gamma_{\text{low}}, \gamma_{\text{high}}) = \sum_{i \in \mathcal{U}_{\text{unpr}}} \mathcal{H}(\gamma_{\text{high}} - \hat{p}_i) \times \mathcal{H}(\hat{p}_i - \gamma_{\text{low}}) \quad (5)$$

Then, "best separate" means to maximize the number of included predictable workers and the number of excluded unpredictable workers, i.e.,

$$\underset{[\gamma_{\text{low}}, \gamma_{\text{high}}] \in [0,1]^2}{\arg\max} \left\{\mathcal{N}_{\text{pred}}^{\text{in}}(\gamma_{\text{low}}, \gamma_{\text{high}}) + \mathcal{N}_{\text{unpr}}^{\text{ex}}(\gamma_{\text{low}}, \gamma_{\text{high}})\right\} \quad (6)$$

We provide an empirical analysis for the error rate thresholds in Section 5.2.

3.2 Workers' utility function

Having developed a method for estimating a worker's error rate, we now determine the weight of each worker in a task assignment. Intuitively, the smaller the error rate is, the larger the utility will be. Next, we provide a discussion about a form of utility which offers a theoretically optimal value of utility based on the error rate.

As anticipated in previous sections, following the lead of similar state-of-the-art models [3,15,21,33], we use majority voting as the fusion method to estimate answers for the task. Considering that workers with different error rate play different roles in majority voting, we assign the answers from different workers with different weights (or utilities). In particular, for the workers whose error rate is higher than 0.5, we assign them negative utilities. This allows us to use also these answers by simply reversing them in the computation of the majority voting. Hence, the estimated answer can be obtained by weighted majority vote

$$h(S_t) = \text{sign}\left(\sum_{i \in S_t} w_i^t Y_{it}\right) \quad (7)$$

where S_t represents the set of selected workers.

Here, we apply the utility design from [27], which claims that the optimal decision rule can be obtained by the weighted majority vote, where

$$w_i^t = \log\left(\frac{1 - p_i^t}{p_i^t}\right), \quad (8)$$

if 1) workers' answers are independent with each other and 2) the probabilities that the ground truth is -1 or 1 is 0.5.

[1]We maintain a sliding window of each error vector with length L for each category. If the new task is in category k, then only the kth sliding window will move forward.

We note that these assumptions are reasonable and fit well with our model. Assumption 1) is realistic, in that most common crowdsourcing sites share a large enough pool of workers to minimize risk of collusion.

As for assumption 2), the probability of a task taking a certain value may vary. Yet, we can "flip" each binary task with probability 0.5 to make the ground truth has equal probability to be 1 or -1. Here, by "flipping a task", we mean that we can modify the binary task such that the ground truth is changed either from 1 to -1, or from -1 to 1. For example, the task "Is website B *suitable* for children?" can be flipped to "Is website B *unsuitable* for children?". In particular, we generate a *flip vector* $\mathbf{h} = [h_1, ..., h_T]$, where each $h_t \in \{-1, 1\}$ $(t = 1, ..., T)$ and $\Pr\{h_t = -1\} = \Pr\{h_t = 1\} = 0.5$. We flip the tasks according to the *flip vector* $\mathbf{h}$: if $h_t = -1$, the task t is flipped; otherwise, the task t is not flipped. Then, the ground truth of each newly generated task vector t is $\overline{Z}_t = Z_t \times h_t$, and $\overline{Z}_t$ has equal probability to be -1 and 1.

4. WORKER SELECTION

In this section, we first formulate the worker selection problem (in Section 4.1), which is defined as a stochastic optimization problem. Due to the hardness of the problem, we replace the objective function with three upper bound functions with the goal of selecting workers that minimize these upper bound functions. Next, we introduce the algorithm at the core of CrowdSelect (in Section 4.2) which is based on observations from the three upper bound functions. Finally, we analyze the time complexity and the theoretical bound of the CrowdSelect algorithm.

4.1 The worker selection problem

In our system, the task requester first posts the tasks in each round. After that, a set of workers, denoted by $\mathcal{U}$ will be asked to complete tasks. The task requester needs to choose some of the workers from $\mathcal{U}$ and coordinate the answers from these workers using some criteria (or fusion method), such as the common *majority voting*, to estimate the correct answer of the task. In the model part, considering the tractability of theoretic analysis, we assume that once a worker agrees to complete a set of tasks, she will not drop them once the task requester selects her. While, in the performance evaluation part (Section 5.3.3), we will analyze how workers' task drop rate will impact the performance of our approach. Note that described here is a single-shot (or single-round) worker selection problem, provided the predicted workers' error rate. Each solution of the worker selection problem is independent, and therefore worker selection problem in case of multiple rounds can be directly extended from the single case.

Suppose that the set of selected workers for task t is S_t $(S_t \subseteq \mathcal{U})$, using the majority voting, the combined answer should be $h(S_t)$ according to Equ. (7).

In return, each worker i receives a payment c_i^t from the task requester. The task requester cannot exceed the budget limit B assigned to the task when requesting answers.

We also use an indicator variable X_{it} to represent whether each worker can correctly classify the task t, where $X_{it} = 1$ indicates the task is correctly classified and $X_{it} = 0$ otherwise. Since we assume workers are independent and not colluding with one another, each X_{it} follows a Bernoulli distribution $B(p_i^t)$, which formalizes the knowledge and uncertainty of worker i towards the task. Without loss of generality, we assume that the correct answer of the task is $+1$, then we can derive that $X_{it} = \frac{1-Y_{it}}{2}$. Then, the probability that the task is incorrectly answered can be represented by

$$\begin{aligned}\Pr\left(\text{sign}\left\{\sum_{i\in S_t} w_i^t Y_{it}\right\} = -1\right) &= \Pr\left\{\sum_{i\in S_t} w_i^t (1 - 2X_{it}) < 0\right\} \\ &= \Pr\left\{\Psi_{S_t} > \frac{\sum_{i\in S} w_i^t}{2}\right\} \quad (9)\end{aligned}$$

where $\Psi_{S_t} = \sum_{S_t} w_i^t X_{it}$.

Lemma 4.1. *Under weighted majority voting, a malicious worker i with error rate p_i^t $(p_i^t > \frac{1}{2})$ and utility w_i^t $(w_i^t < 0)$ can be considered as a reliable worker with probability $1 - p_i^t$ and utility $-w_i^t$.*

Proof. The proof is omitted for lack of space (all proofs will be made available in a complete version of this manuscript) [1]. □

According to Lemma 4.1, before selecting the workers, we can first transfer or "convert" each malicious worker with error rate p_i to a reliable worker with probability $1 - p_i$. After transferring malicious workers to reliable workers, the probability and the utilities of all the workers are no smaller than 0.5 and 0, respectively.

Problem Formulation. Based on the model described above, we formally define the worker selection problem for crowdsourcing tasks as follows:

Instance: Given a binary classification task t, a set of workers $\mathcal{U} = \{1, ..., N\}$, where each worker i has utility w_i^t, cost c_i^t, and probability p_i to correctly answer the task.

Question: How to find a set of workers $S_t \subseteq \mathcal{U}$ to maximize the accuracy of majority voting and the total cost of S_t does not exceed the budget B, i.e.,

$$\min \quad \Pr\left\{\Psi_{S_t} > \frac{\sum_{i\in S_t} w_i^t}{2}\right\} \quad (10)$$

$$\text{s.t.} \quad \sum_{i\in S_t} c_i^t \le B. \quad (11)$$

Here, Ψ_{S_t} is the sum of heterogeneous weighted Bernoulli variables, indicating it follows weighted J-Binomial distribution, which in general has non-polynomial number of items [2]. Considering the computational tractability of the distribution, in what follows, we replace $\Pr\left\{\Psi_{S_t} > \frac{\sum_{i\in S_t} w_i^t}{2}\right\}$ by functions that upper bound it, and then study how to select workers to minimize the upper bound of the error probability. Theorem 4.1 provides three different up bounds of $\Pr\left\{\Psi_{S_t} > \frac{\sum_{i\in S_t} w_i^t}{2}\right\}$.

Theorem 4.1. *The error probability of majority voting from worker set S_t, denoted by $P_{\text{mv}}(S_t)$, is upper bounded by*

1) $\exp\left(-2\left(\frac{\sum_{i\in S_t} w_i^t - 2\mu_{S_t}}{2}\right)^2 / |S_t|\right)$;

2) $\exp\left(-2\left(\frac{\sum_{i\in S_t} w_i^t - 2\mu_{S_t}}{2}\right)^2 / \left(\sum_{i\in S_t} (w_i^t)^2\right)\right)$;

3) $\exp\left(-\left(\frac{\sum_{i\in S_t} w_i^t - 2\mu_{S_t}}{2}\right)^2 / (3\mu_{S_t})\right)$.

Proof. The three bounds are derived by the Hoeffding-Azuma inequality [16], the McDiarmid inequality [26], and the Chernoff inequality [23], respectively. The detailed proof is omitted for lack of space. □

In the following, for simplicity, we respectively name the above three bounds by the *Hoeffding-Azuma bound*, the *McDiarmid bound*, and the *Chernoff bound*.

4.2 A time efficient solution

We develop a time efficient algorithm for the worker selection problem as defined in Equ. (10) and Equ. (11), so as to maximize the accuracy with respect to the budget limit. We first describe the algorithm, and then provide theoretical guarantees of its performance, along with a brief complexity analysis.

4.2.1 Algorithm

The basic idea of the algorithm is based on the results from Theorem 4.1, which tell us that, in order to minimize the bounds, it is essentially to reduce the value of $\sum_{i\in S_t} w_i^t - 2\mu_{S_t}$. Hence the objective function, i.e.,

$$\min\ \Pr\left\{\Psi_{S_t} - \frac{\sum_{i\in S_t} w_i^t}{2} > 0\right\} \tag{12}$$

which is a stochastic optimization problem, and can be simplified to a non-stochastic formula, i.e.,

$$\min \sum_{i\in S_t} w_i^t - 2\mu_{S_t} \tag{13}$$

which also can be written as $\sum_{i\in S_t} w_i^t\left(2p_i^t - 1\right)$. To maximize $\sum_{i\in S_t} w_i^t\left(2p_i^t - 1\right)$, we need to 1) select workers with higher $w_i^t\left(2p_i^t - 1\right)$ and 2) select as many workers as possible, among the ones with lower cost. Considering the above two factors, we define the following metric for each worker i, which is proportional to $w_i^t\left(2p_i^t - 1\right)$ and inversely proportional to his cost c_i^t (line 1-2)

$$a_i^t = \frac{w_i^t(2p_i^t - 1)}{c_i^t}. \tag{14}$$

Here, a_i^t is a relative measure of worker i's error rate to its cost.

Algorithm 1: Pseudo-code for the worker selection algorithm.

```
input  : $p_1^t, ..., p_N^t$, $w_1^t, ..., w_N^t$, $c_1^t, ..., c_N^t$;
output : $S_t$;
1 for $i \leftarrow 1$ to $N$ do
2     $a_i^t \leftarrow \frac{(2p_i^t-1)\times w_i^t}{c_i^t}$;
3 Sort the workers by decreasing $a_i^t$. $B' \leftarrow B$;
4 $S_t \leftarrow \phi$;
5 for $i \leftarrow 1$ to $N$ do
6     if $B' \geq c_i^t$ then
7         $S_t \leftarrow S_t \cup i$; // Put worker $i$ in $S_t$
8         $B' \leftarrow B' - c_i^t$; // The remaining budget is
              decreased by $c_i^t$
9 return $S_t$;
```

After that, we sort the workers by decreasing a_i^t (line 3). Without loss of generality, let the sorted worker sequence be 1, 2, ..., i, ..., N. Finally, we select workers one by one from the sorted sequence. More specifically, we use B' to represent the remaining budget, where B' is initiated by B. In each iteration, we choose item i from the head of the sorted array. If $c_i^t < B'$, we select it and $B' = B' - c_i^t$; otherwise, we skip this item and go to the next one.

4.2.2 Theoretical analysis

We now turn to the theoretical analysis of the performance of the CrowdSelect algorithm. We first provide three theoretical guarantees on the error probability of the CrowdSelect algorithm:

Table 2: The comparison of different upper bounds of CrowdSelect

Conditions		Lowest upper bound
$c_{\min}Nw_{\max}^2 > B$	& $w_{\max}p_{\max} > \frac{1}{6}$	Hoeffding-Azuma
$c_{\min}Nw_{\max}^2 < B$	& $\frac{Bp_{\max}}{Nw_{\max}c_{\min}} > \frac{1}{6}$	McDiarmid
$w_{\max}p_{\max} < \frac{1}{6}$	& $\frac{Bp_{\max}}{Nw_{\max}c_{\min}} < \frac{1}{6}$	Chernoff

Theorem 4.2. *Using CrowdSelect for user selection and weighted majority voting for answer fusion, the error probability from worker set S_t is upper bounded by*

1) the Hoeffding-Azuma bound: $\exp\left(-\frac{c_{\min}\overline{a}^2(B-c_{\max})^2}{2B}\right)$;

2) the McDiarmid bound: $\exp\left(-\frac{\overline{a}^2(B-c_{\max})^2}{2Nw_{\max}^2}\right)$;

3) the Chernoff bound: $\exp\left(-\frac{\overline{a}^2(B-c_{\max})^2c_{\min}}{12Bw_{\max}p_{\max}}\right)$.

PROOF. The detailed proof is omitted for lack of space. □

According to Theorem 4.2, the upper bounds of error probability exponentially decreases as we increase the budget and Table 2 shows the lowest upper bound in three different cases. Now, we improve the error bound above by investigating the performance of CrowdSelect from a PAC (probably approximately correct) perspective. That is, we aim to provide a more efficient bounds which only holds with a high probability. In particular, we state the following:

Theorem 4.3. *Using CrowdSelect for user selection and weighted majority voting for answer fusion, the error probability is upper bounded by the following equation with at least $(1-\beta)$ probability:*

$$\Pr\left(\Psi_{S_t} \geq \frac{\sum_{i\in S_t} w_i^t}{2}\right) \leq -\frac{\overline{a}(B-c_{\max})}{8\sqrt{\frac{B\eta_{\max}}{c_{\min}\beta}}} + \frac{1}{2} \tag{15}$$

where $\eta_{\max} = \max_{i\in S_t}\{(w_i^t)^2\sigma_i^2\}$.

PROOF. The detailed proof is omitted for lack of space. □

It can easily be shown that this PAC bound is lower than the three exponential bounds in Theorem 4.2. In addition, to compare different bounds, we randomly pick up a real task from the Amazon Mechanical Turk trace introduced in Section 5.2, and depict the four different bounds for the task in Figure 2, where $c_{\min}$ =\$0.32, $c_{\max}$ =\$0.89, $\overline{a}$ = 0.43, $p_{\max} = 0.64$, $w_{\max} = 0.90$, $\eta_{\max} = 11.44$, and B is changed from \$1 to \$60. From Figure 2, we can find the PAC bound is always lower than the Hoeffding-Azuma bound, the McDiarmid bound, and the Chernoff bound. Furthermore, when the budget is higher than 15, the error rate of the PAC bound is almost 0. Accordingly, we can guarantee with at least 90% probability that the estimation error is 0 (i.e., we can accurately estimate all the tasks). In addition, the McDiarmid bound is lower than the Hoeffding-Azuma bound and the Chernoff bound when $B \geq 2$, which is consistent with the conclusion shown in Table 2 ($c_{\min}Nw_{\max}^2 < B$ and $w_{\max}p_{\max} > \frac{1}{6}$).

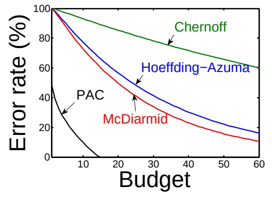


Figure 2: Comparison of different bounds.

Besides weighted majority voting, we can derive the theoretical bounds of CrowdSelect using other fusion methods, like Bayesian classifier [3, 8, 20]. Generally, a fusion method is *weighted majority voting efficient* or (*WMV-efficient* for short) if its outcome is at least as accurate as weighted majority voting given the responses from the same set of workers. More formally, let $P_A(S_t)$ represent the error rate of a fusion method A given the selected worker set S_t, then A is called WMV-efficient iff $P_A(S_t) \leq P_{\mathrm{wmv}}(S_t)$.

Corollary 4.1. *If any WMV-efficient method is applied for answer fusion, the Hoeffding-Azuma, McDiarmid, Chernoff bounds (claimed in Theorem 4.2) and the PAC bound (claimed in Theorem 4.1) still hold for CrowdSelect.*

4.2.3 Complexity of the User Selection Algorithm

It is straightforward to conclude that the proposed algorithm is time efficient according to the following observations.

First, calculating each a_i^t according to Equ. (14) takes one difference, one multiplication, and one division operation. Additions/subtractions require $O(m)$, and multiplications/divisions require $O(m^2)$ single-digit operations [5], where m is the length of symbols to represent the input parameters, e.g., w_i^t, p_i^t, c_i^t, B and etc. Hence, the time complexity for deriving all a_i^t (line 1-2) is $O(m^2N)$.

Second, using typical sorting algorithms, e.g., merge sort [5], calls for $O(N \log N)$ comparison operations, where each comparison takes $O(m)$ single-digit operations [5]. Therefore, the time complexity of sorting a_i^t (line 3) is $O(mN \log N)$.

Third, selecting workers from sorted a_i^t (line 5-8) requires at most N comparisons (between the remaining budget and each worker's cost), hence it takes totally $O(Nm)$ time. Consequently, we can derive that the time compelxity of CrowdSelect is $O(mN \log N + m^2N)$. In reality, the length of m is small, e.g., no more than 10 in the Amazon Mechanical Turk trace. Thus, we consider m as a constant, and the complexity of the user selection algorithm is $O(N \log N)$.

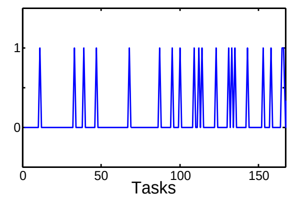
5. EMPIRICAL VALIDATION

In this section, we turn our attention to practical applications of our approach and examine the performance of CrowdSelect in realistic settings. In particular, we run CrowdSelect using data from real-world applications, and compare its performance with a number of alternative methods (in Section 5.3.1). We demonstrate that a) our worker reliability model is effective, and that b) CrowdSelect is superior to the state-of-the-art in terms of accuracy.

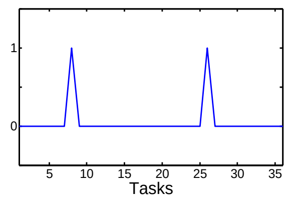
5.1 Datasets

To show the feasibility of the ARMA model and our worker selection strategies, we collected several traces of real world crowdsourcing data. Further, we create our own synthetic dataset as this allows us to freely set the parameters.

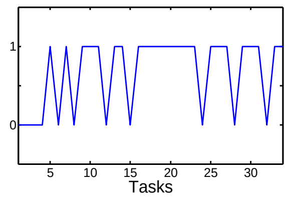
Our real-world dataset includes two sets of traces. First, we collected 4,193 traces from real workers on the popular Amazon Mechanical Turk platform. Workers were asked to read a simple tweet and answer two binary questions, that is: "Does the micro-blog include some geographical information?" and "Does the micro-blog use some humor?" Each Turk task included 20 tweets. Out of the 152 workers, 24 of them were instructed to behave according to a "malicious" strategy. Precisely, they were told to complete the tasks as fast as they could, ignoring accuracy. In these traces 84.2% of the tasks are completed by generally reliable workers, and 15.8% by from malicious workers.



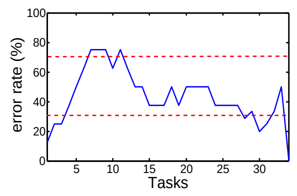
(a) A worker with declining performance and overall 20% error rate



(b) A reliable worker with error rate 6%

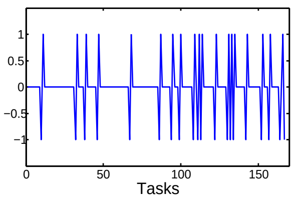


(c) A malicious worker with error rate 67%

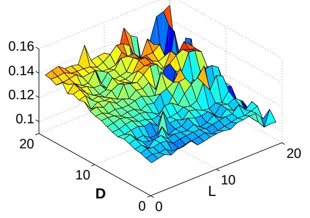


(d) A worker with random performance

Figure 3: Examples of workers' behavior.



(a) Trace from worker in Figure 2(a) after difference



(b) FPE with different L and D

Figure 4: ARMA prediction for one trace of a representative worker.

Second, we also collect 2,200 traces, for a "News" tasks. For these traces, Turk workers were asked to decide whether a given URL was a reliable news site or not. As in the "Twitter" dataset, 24 of them were instructed to behave according to a "malicious" strategy. About 55 workers completed News tasks, wherein 72.8% workers were reliable (i.e., 80% accuracy and above) and 27.2% are malicious. A total of 2,200 traces were generated for the News dataset, with 3.5% of the data subsequently disregarded due to poor quality responses.

Finally, we generate a synthetic dataset as follows. We create 5,000 tasks, where up to 100 workers are available to answer the tasks. We create four types of workers: *reliable workers*, *random workers*, *malicious workers*, and *smart malicious workers*. We assume the first three types of workers follow normal distribution, where their variances equals to 0.1 for all of them, and the means equal to 0.7, 0.5, and 0.3, respectively. Different from normal, malicious workers act so as to give incorrect answers. Finally, smart malicious workers use a strategy to prevent the system to detect their "malicious behavior": when their estimated error rate is high, i.e., 0.7, they try to give correct answer to decrease their error rate. While, when their error rate goes high low, i.e., 0.3, they start to give incorrect answer to deviate the fusion answer. The ratio of the four types of workers is set at 1:1:1:1 by default. Finally, we assume the cost of each worker t, c_i^t, is equal to $1 - p_i^t$.

5.2 Worker Prediction Accuracy

In order to evaluate our accuracy prediction model, we first present some sample workers' behavior from the "Twitter" traces.

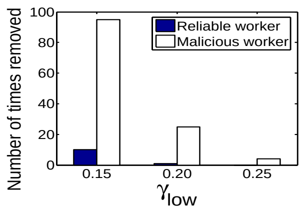
In Figure 3, we present three representative cases of workers response patterns (0 indicates the task is correctly answered and 1 otherwise). The worker in Figure 3 (a) has responses that decay in quality over time. At first the worker's errors are sparse, but they gradually increase in frequency after completing about half the tasks. Figure 3(b) and Figure 3(c) depict the answer patterns from the second and the third workers, respectively. From the figures, the second worker's performance always keeps at a high level (i.e., error rate is below 10%) over time. Hence, we can consider the second worker as a "reliable worker". In contrast, the third worker has high quality of answers at the beginning, i.e., the error rate is only 20% for the first five tasks, but after the 5th task, this worker gives a larger number of incorrect answers (74%). Accordingly, we should consider the third worker as a "malicious worker". Since our approach only considers a worker's most recent behaviors, this malicious worker can be detected since its error rate is around 70% after the 10th task is answered.

Yet, some erratic behavior is hard to detect, since the error rate may fluctuate in a seemingly random manner. For instance, the error rate of worker in Figure 3(d) fluctuates between 20% and 80%. So, we cannot judge whether this worker is likely to give a correct answer in the next round based on her recently completed tasks. To disregard this kind of worker, we must set appropriate *error rate thresholds*. With a high threshold γ_{high} and a low threshold γ_{low} for worker accuracy, e.g., 70% and 30% in Figure 3, this worker will be considered for selection only if its error rate is higher within these bounds for a number of consecutive tasks, e.g., 10 tasks.

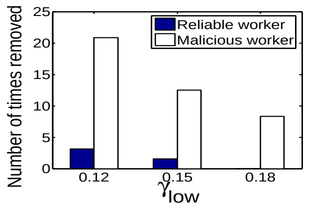
Given a worker's trace, we can predict his error rate by fitting the trace as follows (we specifically use the Box-Jenkins algorithm to fit the ARMA model [6]). Consider the trace of worker in Figure 3(a). Since this original data is not stationary, we need to difference the data, where Figure 4(a) shows the data after differencing. We then change the values of (L, D), estimate the coefficients of ARMA(L, D) for each (L, D) using maximum likelihood estimator (MLE), in which we assume the residuals follow normal distribution.

Figure 4(b) shows the final prediction error (FPE) with varied L and D, where FPE is a criteria to select L and D. Consequently, we find that when $L = D = 5$, the FPE is minimum (FPE = 0.11). Hence, we select ARMA(5,5) as the model for the prediction for this worker.

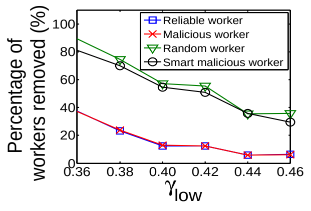
Having presented our ARMA prediction model in one case, we can now analyze the impact of the error rate thresholds, which are used to remove workers with random and erratic behavior. Let $\gamma_{\text{high}} = 1 - \gamma_{\text{low}}$. Through the workers' response from pre-screen tasks, we find that the optimal γ_{low} that can best separate random workers and stable workers in the tweet trace and the News trace should be in $[0.15, 0.25]$ and $[0.12, 0.18]$, respectively. Then, we set γ_{low} by 0.15, 0.2, and 0.25 for the tweet trace, and set γ_{low} by 0.12, 0.15, and 0.18 for the News trace. Figure 5(a) and Figure 5(b) show the number of times that workers are removed by the thresholds in different categories (reliable workers and malicious workers) in the tweet trace and the News trace, respectively. Both figures show that these thresholds are effective



(a) Twitter traces



(b) News traces



(c) Synthetic data

Figure 5: Comparison of removed workers in different categories.

in removing malicious workers, which include workers with erratic behavior, and only few reliable workers are affected.

We further generalize these results by testing our approach on our synthetic dataset. Recall that this dataset also includes the "smart malicious" category of workers, who are more likely to escape detection. Let $\gamma_{\text{high}} = 1 - \gamma_{\text{low}}$. From pre-screen we find that the optimal γ_{low} should be in $[0.36, 0.46]$. Then, in Figure 5(c), we change γ_{low} from 0.36 to 0.46, with 0.02 increase in each step. From the figure we find that the percentage of removed workers is much larger in the categories "random" and "smart malicious" than in the categories "reliable" and "malicious". The random workers are more likely to be removed since their error rate is close to 0.5, which is within the exclusion range between the thresholds. Importantly, the smart malicious workers are also likely to be excluded since when they change their behavior from "reliable" to "malicious" or "malicious" to "reliable", their error rate will approach 0.5, falling in the exclusion range.

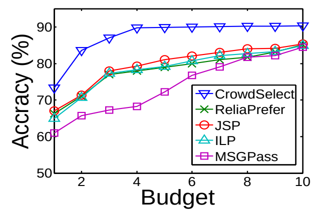
5.3 Empirical Analysis of User Selection

5.3.1 Baseline methods comparison

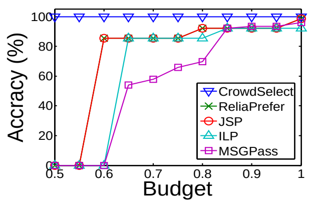
We compare CrowdSelect with the following approaches, which are selected due to their popularity and strong theoretical guarantees.

1. *The Message-Passing algorithm (MSGPass)* proposed by Karger *et al.* [18], [17]. MSGPass considers the task assignment as a process of designing a bipartite graph, where each edge corresponds to a task-worker assignment. To generate the bipartite graph, MSGPass uses a random graph generation scheme called the *configuration model* [4].

2. *Integer Linear Programming based method (ILP)* proposed by Ho *et al.* [15]. ILP formulates the task assignment as an ILP problem, whose objective is to 1) minimize the cost of hiring workers and meanwhile 2) guarantee the the estimated answer's error probability is lower than a predetermined threshold. In the following, to better compare ILP with CrowdSelect, we modify the objective of ILP to be consistent with CrowdSelect, i.e., to 1) minimize the error probability



(a) Tweets traces



(b) News traces

Figure 6: Comparison of different strategies using weighted majority voting (real trace).

of majority voting and 2) make sure the cost is within the given budget limit.

3. *Jury Selection Problem (JSP)* [7]. This method models the user selection process in crowdsourcing as the well known Jury selection problem, where the majority answer is selected as the true answer. In particular, [7] proposes a heuristic algorithm to solve the problem.
4. *Reliability Preferred (ReliaPrefer)*. This simple baseline method selects the answers from the workers with higher reliability and lower cost. In particular, ReliaPrefer first sorts all the workers in descending order of the ratio of reliability and cost, and then selects the workers one by one from the sorted order.

Notice that both MSGPass and ILP only consider workers with error rate lower than or around 0.5 [15, 17, 18], hence, in the following, we remove malicious workers when implementing these two methods. To summarize the key differences between CrowdSelect and the compared methods, we highlight the main features of all these methods in Table 3.

Table 3: The comparison of different algorithms

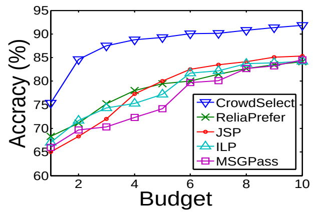
	Malicious workers' answers are included	Error rate is considered	Cost is considered
CrowdSelect	√	√	√
ILP		√	√
ReliaPrefer	√	√	
JSP		√	√
MSGPass			

5.3.2 Real-world data evaluation

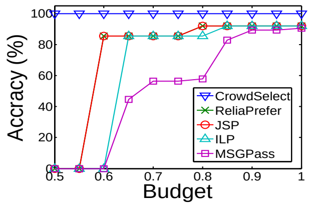
Real-data eveluation is carried out using both the Twitter dataset and the "News" traces, described in section 5.1.

Figure 6(a) and Figure 6(b) compare the accuracy of different approaches using the tweets trace and the News trace, respectively. From both figures, we observe that CrowdSelect has consistently higher accuracy than other strategies: 1) CrowdSelect is better than ILP and ReliaPrefer because CrowdSelect takes advantage of malicious workers; 2) CrowdSelect outperforms MSGPass since it selects workers providing more information, i.e., the error rate is close to either 0 or 1, to increase the accuracy, while MSGPass ignores the diversity of workers' error rate and gives each worker equal probability to be selected.

To demonstrate that CrowdSelect outperforms other methods not only using weighted majority voting, but also other classifiers, we also compare different worker selection strategies using Bayesian classifier, which is also a widely used fusion method to combine answers from selected workers [3, 8, 20]. Specifically, given the prior distribution over the responses -1 and 1, which has the same probability as mentioned in Section 3.2, and the estimated error rate of each selected worker, we can obtain the marginal posterior

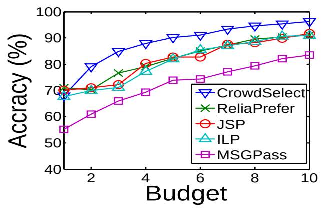


(a) Tweets traces

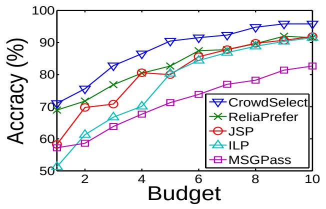


(b) News traces

Figure 7: Comparison of different strategies using the Bayesian classifier (real trace).



(a) Bayesian classifier



(b) Weighted majority voting

Figure 8: Comparison of different strategies using synthetic data.

distribution over -1 and 1 using Bayes' Formula. To minimize the probability of error we choose the response with the maximum posterior for each item as the aggregated response. Figure 7(a) and Figure 7(b) depict the accuracy of different strategies from both traces, from which we can observe that the results are consistent with Figure 6(a) and Figure 6(b).

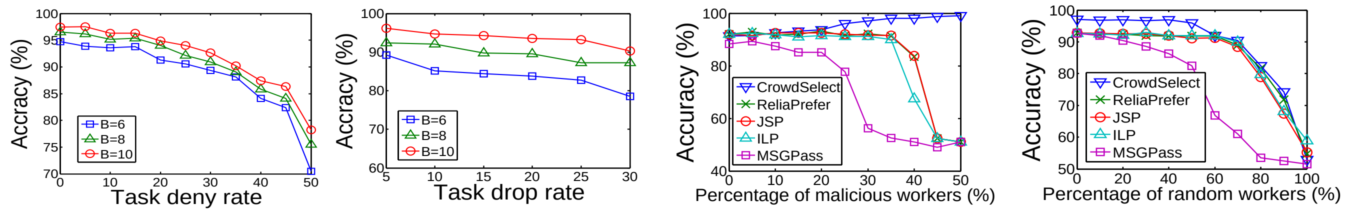
5.3.3 Synthetic data evaluation

We next evaluate the performance of CrowdSelect through simulations on synthetically generated data.

We first compare the accuracy of our CrowdSelect approach with our baseline methods, using both weighted majority voting and Bayesian classifier in Figure 8(a) and Figure 8(b), respectively. From both figures, we observe that CrowdSelect outperforms ILP, JSP, ReliaPrefer, and MSGPass, since CrowdSelect takes advantage of malicious workers, who have high informative answers and low cost. In addition, we find that CrowdSelect, ILP, JSP, and ReliaPrefer are superior to MSGPass, since CrowdSelect, ILP, and ReliaPrefer select workers with lower error rate to increase the accuracy, while MSGPass selects workers randomly.

After comparing CrowdSelect with different approaches, we now change different parameters to perform a thorough study for CrowdSelect's performance. In particular, we first change workers' *task deny rate*, i.e., the probability that a worker denies a task, in Figure 9(a), from which we can find that the accuracy decreases with the increase of the task deny rate. This is because that, on average, higher task deny rate indicates less workers provided to be selected for each task, and hence decrease the accuracy. In Figure 9(b), we change the workers' *task drop rate*, which is defined as the probability that a worker drops a task after accepting it. Not surprisingly, the figure shows that the accuracy decreases with the increase of the task drop rate, since the more workers drop the task after being selected, the less answers the task requester will collect, which consequently decreases the accuracy.

Recall that as shown in Figure 6-8, CrowdSelect outperforms previous methods since it takes advantage of malicious



(a) changing workers' task deny rate

(b) changing workers' task drop rate

(c) changing malicious worker percentage

(d) changing random worker percentage

Figure 9: Synthetic data simulation with parameter changes.

workers. Based on this observation, we aim to investigate how the distribution of reliable workers, malicious workers, and random workers might affect the performance of CrowdSelect. To this end, we first create a scenario, where the percentage of both random workers and smart malicious workers are set by 25%, and the percentage of malicious workers is change from 0% to 50% (correspondingly, the reliable workers' percentage is changed from 50% to 0%). The accuracy of different strategies under this scenario is depicted in Figure 8(c). Different from all the compared methods whose accuracy decreases when the percentage of malicious workers increases, the accuracy of CrowdSelect goes up with the increase of malicious workers. It is because that, in CrowdSelect, malicious workers can be used as reliable workers, but have less cost than reliable workers. The figure also demonstrates that CrowdSelect has at least 17.5% gain in term of accuracy compared to all the compared methods, when more than 50% workers are malicious (including smart malicious workers).

We also change the percentage of random workers in the worker pool from 0% to 100%, and maintain the ratio among reliable workers, malicious workers, and smart malicious workers by 1:1:1. The accuracy of different strategies under this scenario is depicted in Figure 8(d). As expected, the accuracy of all the methods goes down when the percentage of random workers increases, since the behavior of random workers is difficult to capture, which hinders the task requester from benefiting from multiple labels.

6. RELATED WORK

Repeated labeling has attracted considerable attention, dating back to the expectation-maximization (EM) based algorithm proposed by Dawid and Skene [12]. Since then, much work has followed up by modifying and extending this approach in different aspects [7, 8, 11, 24, 25, 29–32, 35–38]. For example, Whitehill et al. [36] and Welinder et al. [35] have applied the EM algorithm to more complicated probabilistic models for image labeling tasks. In a similar vein, a large body of work uses Bayesian learning techniques [8, 24] to predict workers' responses and detect spammers, i.e., workers submitting arbitrary answers independent of the question in order to collect their fee, such as [29, 30]. Dai et al. used partially observable Markov decision process to model the estimation's quality [10].

On the theoretical side, many efforts have been devoted to provide theoretical guarantees of the estimation error of answers. For example, Ghosh et al. [14] presented a spectral algorithm that provably learns the true item qualities, with bounded error. Karger et al. [18] used belief propagation to derive both a set of worker reliabilities and an estimate for item qualities for a sparse random graph. Liu et al. [24] extended the BP algorithm of [18] via a Bayesian approach by choosing a suitable prior for item qualities and worker reliabilities, and uses clever techniques to make the message passing more efficient.

However, the above works mainly focus on how to aggregate the labels from multiple labelers, where the selected workers are given. The closest related works to ours are from Cao et al. [7], Zhao et al. [38], Karger et al. [17,18] and Ho et al. [15]. Cao et al. [7] modeled the worker selection process as the well-known Jury Selection Problem, in which they aim to select a jury from a set of candidate jurors to maximize the accuracy of majority voting, where each candidate juror is associated with a payment and error rate. In particular, they assume that all the jurors (workers) have the same weight in majority voting. Zhao et al. [38] proposed a natural framework for capturing the latent skills of workers as well as the latent categories of crowdsourced tasks. Similar to our work, they estimated the latent skills of each worker based on the worker's historical record. Karger et al. introduced a model in which a requester has a set of homogeneous labeling tasks he must assign to workers who arrive online. They proposed an assignment algorithm based on random graph generation and a message-passing inference algorithm inspired by belief propagation, and showed that their technique is order-optimal in terms of labeling budget. Ho et al. [15] generalized this model to allow heterogeneous tasks, so that the probability that a worker completes a task correctly may depend on the particular task. As shown in our empirical comparison reported in Section 5.3.3, despite these similarities our approach is superior. CrowdSelect obtains stronger accuracy guarantees, as it adopts a more realistic model embracing workers who may adopt any possible behavior.

Finally, Kobren et al. [19] presented dynamic approaches for task allocation and goal deployment to promote long-term worker engagement. Although related, the objective of [19] is different from ours, i.e., it aims to improve long-term workers' participation, while our goal is to improve the estimated answer for the task requester given through selecting worker from a fix number of workers.

7. CONCLUSION

In this paper, we presented Crowdselect, a time efficient task assignment strategy accounting for workers error rate as they complete tasks and display a certain bias of behavior. We demonstrate that CrowdSelect outperforms state-of-the-art methods using both real-world and synthetic data.

In the future, we plan to extend our models to offer stronger guarantees, with respect to both accuracy and performance. In particular, with respect to performance, we plan to investigate how to update the assignments as workers leave the system or new ones are added. Finally, CrowdSelect is currently only suitable for binary tasks. A natural extension is to support non-binary tasks. A major challenge for non-binary tasks is to accurately anticipate the error rate of workers, and calculate workers' probability accordingly.

8. REFERENCES

[1] Crowdselect: Increasing accuracy of crowdsourcing tasks through behavior prediction and user selection. https://www.dropbox.com/s/0v152ar3myujouv/CIKM_full.pdf?dl=0.

[2] J. Benneyan, B. Harris, and A. Taseli. Applications and approximations of heterogeneous weighted and unweighted j-binomial probability distributions. In *Proc. of CIE*, 2007.

[3] D. Berend and A. Kontorovich. A finite sample analysis of the naive bayes classifier. *Journal of Machine Learning Research*, 2008.

[4] B. Bollob. *Random Graphs*. Cambridge University Press, 2001.

[5] J. M. Borwein and P. B. Borwein. *Pi and the AGM: A Study in the Analytic Number Theory and Computational Complexity*. Wiley-Interscience, 1987.

[6] P. Brockwell and R. Davis. *Introduction to Time Series and Forecasting*. Springer, 1996.

[7] C. Cao, J. She, Y. Tong, and L. Chen. Whom to ask? jury selection for decision making tasks on micro-blog services. In *Proc. of VLDB*, 2012.

[8] B. Carpenter. A hierarchical bayesian model of crowdsourced relevance coding. In *Proc. of TREC*, 2011.

[9] Crowdflower. http://www.crowdflower.com/.

[10] P. Dai, Mausam, and D. S. Weld. Artificial intelligence for artificial artificial intelligence. In *Proc. of AAAI*, 2011.

[11] M. Davtyan, C. Eickhoff, and T. Hofmann. Exploiting document content for efficient aggregation of crowdsourcing votes. In *Proc. of CIKM*, 2015.

[12] P. Dawid and A. Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Journal of the Royal Statistical Society*, 1979.

[13] J. S. Downs, M. B. Holbrook, S. Sheng, and L. F. Cranor. Are your participants gaming the system?: Screening mechanical turk workers. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, 2010.

[14] A. Ghosh, S. Kale, and R. P. McAfee. Who moderates the moderators?: Crowdsourcing abuse detection in user-generated content. In *Proc. of EC*, 2011.

[15] C.-J. Ho, S. Jabbari, and J. W. Vaughan. Adaptive task assignment for crowdsourced classification. In *Proc. of ICML*, 2013.

[16] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of American Statistical Association*, 1963.

[17] D. Karger, S. Oh, and D. Shah. Budget-optimal task allocation for reliable crowdsourcing systems. In *CoRR*, 2011.

[18] D. Karger, S. Oh, and D. Shah. Iterative learning for reliable crowdsourcing systems. In *Proc. of NIPS*, 2011.

[19] A. Kobren, C. H. Tan, P. Ipeirotis, and E. Gabrilovich. Getting more for less: Optimized crowdsourcing with dynamic tasks and goals. In *Proc. of WWW*, 2015.

[20] A. Lacasse, F. Laviolette, M. Marchand, P. Germain, and N. Usunier. PAC-Bayes bounds for the risk of the majority vote and the variance of the Gibbs classifier. In *Proc. of NIPS*, 2006.

[21] F. Laviolette, M. Marchand, and J.-F. Roy. From pac-bayes bounds to quadratic programs for majority votes. In *Proc. of ICML*, 2011.

[22] M. Lease. On quality control and machine learning in crowdsourcing. *Human Computation*, 11:11, 2011.

[23] X. Lin, C. Genest, D. L. Banks, G. Molenberghs, D. W. Scott, and J.-L. Wang. *Past, Present, and Future of Statistics*. CRC Press, 2014.

[24] Q. Liu, J. Peng, and A. Ihler. Variational inference for crowdsourcing. In *Proc. of NIPS*, 2012.

[25] X. Liu, M. Lu, B. C. Ooi, Y. Shen, S. Wu, and M. Zhang. CDAS: A crowdsourcing data analytics system. In *Proc. of VLDB*, 2012.

[26] C. McDiarmid. On the method of bounded differences. *Surveys in Combinatorics*, 1989.

[27] J. Neyman and E. S. Pearson. On the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society A*, 1933.

[28] D. Oleson, V. Hester, A. Sorokin, G. Laughlin, John, Le, and L. Biewald. Programmatic gold: Targeted and scalable quality assurance in crowdsourcing. In *Proc. of HCOMP*, 2011.

[29] V. C. Raykar and S. Yu. Eliminating spammers and ranking annotators for crowdsourced labeling tasks. *JMLR*, 2012.

[30] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy. Learning from crowds. *JMLR*, 2010.

[31] M. Rokicki, S. Chelaru, S. Zerr, and S. Siersdorfer. Competitive game designs for improving the cost effectiveness of crowdsourcing. In *Proc. of CIKM*, 2014.

[32] 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 *Proc. of EMNLP*, 2008.

[33] L. Tran-Thanh, M. Venanzi, A. Rogers, and N. R. Jennings. Efficient budget allocation with accuracy guarantees for crowdsourcing classification tasks. In *Proc. of AAMAS*, pages 21–26, 2013.

[34] A. M. Turk. https://www.mturk.com/mturk/welcome.

[35] P. Welinder, S. Branson, S. Belongie, and P. Perona. The multidimensional wisdom of crowds. In *Proc. of NIPS*, 2010.

[36] J. Whitehill, P. Ruvolo, T. Wu, J. Bergsma, and J. Movellan. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Proc. of NIPS*, 2009.

[37] J. Zhang, S. Sheng, J. Wu, X. Fu, and X. Wu. Improving label quality in crowdsourcing using noise correction. In *Proc. of CIKM*, 2015.

[38] Z. Zhao, F. Wei, M. Zhou, W. Chen, and W. Ng. Crowd-selection query processing in crowdsourcing databases: A task-driven approach. In *Proc. of EDBT*, 2015.