Practical POMDP-based test mechanism for quality assurance in volunteer crowdsourcing

Peng Shi, Wanyuan Wang, Yifeng Zhou, Jiuchuan Jiang, Yichuan Jiang, Zhifeng Hao & Jianyong Yu

To cite this article: Peng Shi, Wanyuan Wang, Yifeng Zhou, Jiuchuan Jiang, Yichuan Jiang, Zhifeng Hao & Jianyong Yu (2018): Practical POMDP-based test mechanism for quality assurance in volunteer crowdsourcing, Enterprise Information Systems, DOI: 10.1080/17517575.2018.1493539

To link to this article: https://doi.org/10.1080/17517575.2018.1493539

Published online: 09 Jul 2018.

Submit your article to this journal

Article views: 6

View Crossmark data
**ABSTRACT**

In volunteer crowdsourcing, tasks are published via an open call and completed by many workers without reward. Under the traditional volunteer crowdsourcing paradigm, workers with diverse levels of reliabilities are chosen indiscriminately; moreover, each worker’s performance may change over the time. Thus, the quality of task completions is a key concern in volunteer crowdsourcing. To improve the task completion quality (i.e. the accuracy of task answers), we adopt an adaptive test task (with a true answer) insertion approach to detect a worker’s performance dynamically, thereby ensuring that normal tasks (with unknown true answers) are assigned when this worker is currently deemed reliable via testing. To decide when to route test tasks to detect a worker’s performance or assign normal tasks to be completed in a high quality state, we proposed a Partially Observable Markov Decision Processes (POMDP) based test mechanism without any complicated parameter estimation, which is more practical for real-world volunteer crowdsourcing applications. In addition, we also designed rejection strategies to reject malicious workers and dubious answers. Experiments on real datasets demonstrate that the proposed test mechanism performs better in the accuracy of task answers, compared with benchmark methods.

**1. Introduction**

Crowdsourcing was first proposed by Howe (2006) and has become a popular business model (Xiong, Zhang, and Zhu 2017) that can harness collective human intelligence to perform tasks that are hard to be solved by computers (Ho, Jabbari, and Vaughan 2013), such as image annotation (Tran-Thanh et al. 2013), and text translation (Dai, Lin, and Weld 2013). Public or private and profit or non-profit organizations all can utilize crowdsourcing as a problem-solving paradigm (Wang et al. 2017a; Wu et al. 2017; Kaufmann, Schulze, and Veit 2011). Specifically, crowdsourcing under a volunteer way, due to a large quantity of free workers, is more suitable for public projects. In recent years, some well-known volunteer crowdsourcing platforms have appeared (Mao et al. 2013b), such as Tomnod and Galaxy Zoo (Lintott et al. 2008). For example, a typical application of volunteer crowdsourcing is that Tomnod called thousands of Internet users to classify and label damaged roads, buildings and bridges after Nepal Earthquake, to help the ground aid.
In volunteer crowdsourcing paradigm, tasks are published via an open call and completed by a large number of voluntary workers without reward (Mao et al. 2013a). Although many free workers can greatly reduce overhead, compared to professional workers, volunteer workers’ reliabilities are unknown. In practice, the performance of a worker may vary over the time (Jung, Park, and Lease 2014). For instance, when a worker is distinguishing the galaxy type (elliptical or spiral) in the image at Galaxy Zoo, the worker’s accuracy may decrease due to fatigue or boredom (Bragg et al. 2016) and increase due to experience accumulation (Carterette and Soboroff 2010) or when this worker’s concentration returns. Thus, task completion quality is a key challenge that should be highly concerned (Dawid and Skene 1979; Ipeirotis, Provost, and Wang 2010; Karger, Sewoong, and Shah 2011). For the binary option tasks, the accuracy that denotes the ratio of correct answers is usually adopted to measure the task completion quality (Jung and Lease 2015; Bragg and Weld 2016). Furthermore, without the profit supply and strict behavior constraints, the number of task completions for a voluntary worker is arbitrary. Traditional quality detection models often adopt a fixed periodic policy or require a limited-length initial training phrase, which is not actively adaptive to unstable number of task completions and variable worker accuracy that can be up or down (Bragg and Weld 2016; Jung and Lease 2015). Besides, complex update-based models do not apply to the crowdsourcing system with large-scale volunteer workers’ participations. To improve the quality of completed tasks, we consider how to build a practical test model which is adaptive to the temporal fluctuation in worker’s performance.

To solve above problems, we adopt an adaptive test task (with true answer) insertion approach to detect a worker’s reliability dynamically, thereby ensuring that normal tasks (with unknown true answers) are assigned when this worker is currently in a reliable state. A worker’s performance can be easily estimated after performing a test task, because the true answer of the test task is known. As soon as the worker’s state becomes better, normal tasks can be rerouted to this worker. Even when the worker currently has a bad performance but is not malicious, doing test tasks can avoid unreliable answers to influence the final results of tasks, resembling filtration of bad answers; on the other hand, test tasks are also beneficial to sense an increase in a worker’s future performance. However, if the worker’s accuracy is lower than the threshold, the worker should also be rejected by the crowdsourcing system. In this paper, our test mechanism is to detect a worker’s current performance and decide when to route a test task or a normal task to the worker for improving the task completion quality. Our testing mechanism relies on embedded test tasks to ensure the quality of task completions, therefore, the mechanism should apply to the crowdsourcing scenario where the number of tasks is not very large, and many workers can involve in the task execution, but there is a certain requirement on the task completion quality.

Our main contributions can be summarized as: 1) considering the fluctuation (up or down) of worker’s performance over time and the quality of task completions, we carefully design and build a practical POMDP-based decision model by embedding test tasks to detect a worker’s current performance; 2) cooperating with the running of POMDP model, we design an observation maker based on a predictor to provide proper observations; 3) rejection strategies are presented to ignore the malicious worker whose accuracy is lower than the threshold, and dubious answer with lower observation maker's confidence; 4) because of careful designed decision model, our Test Mechanism excludes any complicated update process or parameter estimation, which is feasible for real volunteer crowdsourcing and beneficial for higher capacity and lower system load; and 5) we conduct rich experiments on real datasets to evaluate our method’s performances, and experimental results show that comparing with prior work, our mechanism outperforms its rivals in the accuracy of task answers and can complete more tasks.

The rest of the paper is organized as follows. The second section reviews prior work. Section 3 in detail introduces the POMDP-based adaptive test mechanism with the modules of observation maker and rejection strategies, followed by experiments on real datasets in Section 4. In addition, the last section provides conclusion and future work.
2. Related work

The quality of completed tasks is a primary concern due to the unreliability of the unknown workers (Alotaibi and Liu 2017). Sheng, Provost, and Ipeirotis (2008) stated that repeated-labeling can improve both the quality of the labeled data and trained model when labeling is noisy. Whitehill et al. (2009) introduced a probabilistic model to simultaneously infer the label of each image, the expertise of each labeler, and the difficulty of each image. Some works are devoted to assigning tasks to good workers to improve the quality (Karger, Sewoong, and Shah 2011, 2014; Rajpal, Goel, and Mausam 2015), simultaneously balancing exploration and exploitation of workers’ abilities (Donmez, Carbonell, and Schneider 2009; Tran-Thanh et al. 2013; Ho, Jabbari, and Vaughan 2013). There are also some works that focus on designing payments to incentive workers (Wu et al. 2017; Yin and Chen 2015). Considering that crowd workers are in social network (Cao et al. 2016; Bu et al. 2017), Wang et al. (2017a) focused on team formation for complex crowdsourcing tasks. From a novel point-of-view to improve the quality, Wang et al. (2017b) learned to distinguish between easy and hard tasks for workers. Above all, these works assume that workers are in a stable probability to give correct answers. Recent work shows that in practice a worker’s behavior can change dynamically, and the reliability of each worker varies over the time (Jung, Park, and Lease 2014). Donmez, Carbonell, and Schneider (2010) proposed a framework based on Sequential Bayesian Estimation to estimate each worker’s accuracy at each time step and decide which annotators to be queried. However, execution history and complex calculations increase the system load, and all records of execution may be not available due to some anonymous voluntary workers. Jung, Park, and Lease (2014) proposed a time-series label prediction model for crowdsourced work but assumed strongly that all examples’ gold labels are available immediately, which is not appropriate for unknown labeled tasks (Bragg, Mausam, and Weld 2014). In real volunteer crowdsourcing, there are usually a large number of workers from all over the world, so it is not accessible to complex computation. Therefore, to improve the quality of task completions, we propose a simple method, based on test tasks, for detecting each worker’s performance and assigning normal tasks under the worker’s good performance state, in accordance with proper rejection strategies. This method decides when to route a test task or a normal task to a worker, via an optimal POMDP policy that can be calculated offline.

Using static test tasks to estimate a worker’s quality is common in crowdsourcing field. Jung and Lease (2015) considered how to estimate each worker’s performance modeled by temporal correlation with limited gold labels. However, gold labels are placed on the initial phase or used to check label correctness periodically. These static strategies are not actively adaptive to the fluctuation of crowd workers’ performance. Besides, fixed or periodic strategies have the potential risk that the test frequency is mastered by workers. Therefore, there is a need to guide adaptive test task insertion. Due to the variance in reliability and performance of workers over the time, and no payment limitation in volunteer crowdsourcing, intuitively, workers in a bad state should continuously perform test tasks for two reasons: 1) doing test tasks in a bad state cannot influence the aggregation of normal tasks; 2) doing test tasks can sense the rising up of a worker’s future performance and route normal tasks again. The static test strategies cannot implement these functions.

There is a related prior work of Bragg et al. (2016), who also use POMDP model to decide actively when to have a test and whether to fire the worker if he or she does not reach to target accuracy supplied by the requester. There are four main differences with our work as follows: 1) they assume that a worker’s accuracy can only decrease over time, and when the accuracy of a worker is lower than the target of the requester, the worker is fired. Our work is based on a generalization of this assumption, because a worker’s accuracy may decrease due to fatigue or boredom (Bragg and Weld 2016) and increase due to experience accumulation (Carterette and Soboroff 2010) or returned concentration; 2) their solution balances testing workers to estimate the worker accuracy and actually getting tasks performed by good workers. Upon volunteer...
crowdsourcing, the cost of the test task will not be focused on, so this balance in our work is converted to a trade-off between sensing the current accuracy and re-assigning normal tasks to workers; 3) their model needs five parameters to be learned through a HMM model and an EM algorithm, and their POMDP model should be updated periodically because of parameter re-estimation. Our model with no parameter to be estimated, is beneficial for system load reduction; 4) their method achieves efficient results but relies on a basic policy at the initial phase which explores parameters. Our model can be adaptive for short- or long-term task execution, because there is no fixed phase for initialization.

3. Model and algorithm

In volunteer crowdsourcing, a worker’s performance varies over time, up or down. This paper focuses on the binary option tasks, e.g. distinguishing the galaxy type (elliptical or spiral) in the image, and each task only has one correct answer. Test task is defined as the true answer known task, and the normal task with unknown true answer needs to be completed by workers. Therefore, we adopt the accuracy of normal task answers to measure the task completion quality. To improve the task completion quality, our methodology is to route normal tasks to the worker when this worker is currently in a good state because he has a high probability of giving the right answers, and conversely assign test tasks to the underperforming worker to wait for the recovery of the performance. There is possible to get enough test tasks in volunteer crowdsourcing, because normal tasks (with unknown true answers) can be converted to answer-known test tasks according to expert labels or prior worker label aggregation such as Majority Voting (Sheng, Provost, and Ipeirotis 2008) and the EM algorithm (Whitehill et al. 2009). In the whole process, workers should not notice the differences between test tasks and normal tasks, and all workers should fulfill the tasks one by one as Figure 1. This problem is formulated as a decision-making problem about what the next task type (test task or normal task) will be assigned according to the worker’s past answer and task type. The decision maker, who decides when to route a test or normal task to a worker, is called a controller agent. For example, the controller agent sends a test task to the worker according to its strategy, and the worker performs the task and submits her answer; comparing the worker’s answer with the task’s true answer, the controller agent can know the worker’s answer is correct and decide to next send a normal task to this worker.

In the model and algorithm, our test mechanism is composed by the three module as Figure 2: POMDP-based decision model, the observation maker and rejection strategies. Due to the unobservability of worker states and normal tasks’ true answers, we use POMDP to model this decision process. To cooperate with the running up of the process, an observation maker is designed to provide a reasonable judgment about observations. Based on the information provided by above two modules, rejection strategies are proposed to refuse malicious workers and unreliable labels.

The original contribution can be summarized as below. POMDP is a general model that makes sequential decisions under uncertain environment, but how to design the six-tuple \(< S, A, O, T, P, R >\) for the practical problem is the key to the model’s ability to achieve good results in real world. Our first

![Figure 1. Flow chart for the system.](image-url)
3.1. Basic model based on POMDP

In this paper, we consider binary option tasks and the difficulty of all tasks is consistent. There are only two types of tasks, normal and test task. Normal tasks need to be completed, and their true answers are unknown. In contrast, test tasks’ true answers are certain. Additionally, answer-known test tasks have two main functions: ‘sensor’ and ‘filter’: given the real truth, test tasks can easily detect a worker’s performance by judging the worker’s answer; if the current performance of a worker is bad but the worker is not malicious, we should continue to assign test tasks and would not be anxious to give up this worker; because there is possible that this worker can do better in the future and the assigned test tasks can be used as probes that can sense the worker’s improvement in performance; Even if the worker is in a bad state and performs test tasks, the unreliable answers cannot make a negative influence on the normal tasks’ results; in other words, test tasks can prevent the worker generating unreliable answers for normal tasks, just like using a sieve to filter out the impurities.

POMDP (Partially Observable Markov Decision Processes) is a flexible model that makes sequential decisions under the uncertain scenario where the agent cannot directly observe its environmental states (Cassandra, Kaelbling, and Littman 1994; Rajpal, Goel, and Mausam 2015). However, observations caused by actions can supply the agent with useful information. The optimal policy of
POMDP is to find an action function under observations, which can maximize the rewards in the overall decision processes. In our problem, when a worker performs a normal task with an unknown true answer, we cannot directly observe the current state of the worker. Thus, we adopt a typical POMDP scheme to model the sequential decision process and decide when to route a test or normal task according to the optimal policy.

A POMDP model is defined by a six-tuple \((S, A, O, T, P, R)\):

- **\(S\)**: is a finite set of environmental states.
- **\(A\)**: is a set of all actions.
- **\(O\)**: is a set of all observations.
- **\(T\)**: is the state transition model, describing the probability from the current state to another state after an action.
- **\(P\)**: is the observation function for the probability of receiving an observation from a state after an action.
- **\(R\)**: is the reward function of receiving an observation from a state after an action.

Based on the intuitive ideas above, we introduce in detail how to map this problem to a POMDP model as follows.

### 3.1.1. States

A state \(s\) can be represented by \((q, t)\), where \(q\) is the quality level of a worker and \(t\) is the work status which can switch between **Test** and **Normal**. To avoid the complexity of continuous POMDP, we assume that \(q\) is the discrete variable to represent each worker’s current accuracy. For example, \(Q = \{0.2, 0.4, 0.6, 0.8\}\) and \(q \in Q\). In addition, \(t\) indicates the status of work mode, \(T = \{Test, Normal\}\), \(t \in T\). In **Test** mode, the worker performs test tasks, where the correctness of answers can be directly observed. Similarly, the worker does normal tasks when in **Normal** mode. For instance, the state \((0.8, Normal)\) denotes that the worker’s current accuracy is 0.8 and the worker is doing a normal task. A terminate state is not needed in this model, because our intuitive idea uses test tasks to prevent the worker generating unreliable answers for normal tasks when the worker is in a bad state. Limited state space is beneficial to simplify the solving process of the optimal policy.

**Example 3.1.** We assume \(Q = \{0.2, 0.4, 0.6, 0.8\}\) and \(T = \{Test, Normal\}\), thus, the states are \(S = \{(0.2, Test), (0.4, Test), (0.6, Test), (0.8, Test), (0.2, Normal), (0.4, Normal), (0.6, Normal), (0.8, Normal)\}\).

### 3.1.2. Actions

In our model, there are only two actions that should be considered: **test** and **assign**. When the worker performs well, it is a good time to **assign** this worker to normal tasks. For the volunteer crowdsourcing, there are a large number of free workers contributing to the projects, so the test task costs are tolerant to be considered. Test tasks can be used to sense the rise of a worker’s accuracy for the future assignment of normal tasks. Moreover, test tasks are also able to prevent the worker generating unreliable answers for normal tasks. Therefore, the above two actions are enough, and the action, **rejecting** a worker, is not necessary. For the worker who has an adversarial intent or when the worker’s accuracy is below a certain value, the rejection strategies will take effects.

### 3.1.3. Transitions

\(T\) is defined as the state transition probability. Specifically, \(T(s' | s, a)\) denotes the probability that taking an action \(a\) in a state \(s\) will result in another state \(s'\). The state \(s\) is constituted by \((q, t)\) where \(q\) denotes the worker’s current ability level and \(t\) indicates the current status of work mode, \(t \in \{Test, Normal\}\). The two action, **test** and **assign**, can determine the worker’s next work mode (performing a test task or normal task), and obviously the worker cannot be simultaneously in two work mode. Thus, the actions can only directly cause the switch between the two work mode (**Test** or **Normal**) but
cannot affect transition among different capacity level \( q \). In other words, after taking an action (test or assign), the work mode \( t \) in the state \( s \) may transform but the ability level \( q \) remains the same. Therefore, the transition probability is defined as Eq.(1)-(4) and the probability of transition between any two states with different ability levels is 0.

\[
T(<q, \text{Test}> | <q, \text{Test}>, \text{test}) = 1.0, \ \forall q \in Q
\]  

(1)

\[
T(<q, \text{Test}> | <q, \text{Normal}>, \text{test}) = 1.0, \ \forall q \in Q
\]  

(2)

\[
T(<q, \text{Normal}> | <q, \text{Test}>, \text{assign}) = 1.0, \ \forall q \in Q
\]  

(3)

\[
T(<q, \text{Normal}> | <q, \text{Normal}>, \text{assign}) = 1.0, \ \forall q \in Q
\]  

(4)

Example 3.2. The transition probabilities \( T(<0.6, \text{Test}> | <0.6, \text{Test}>, \text{test}) \) and \( T(<0.6, \text{Test}> | <0.6, \text{Normal}>, \text{test}) \) are 1.0, but the transition probabilities \( T(<0.8, \text{Test}> | <0.6, \text{Test}>, \text{test}) \) and \( T(<0.6, \text{Normal}> | <0.6, \text{Test}>, \text{test}) \) are 0.0.

In the POMDP model, a belief state \( b \) is usually used as an internal state to denote a probability distribution of all the environmental states \( S \) (Cassandra, Kaelbling, and Littman 1994). To facilitate presentation, we should number each state \( s_i = \langle q, t \rangle \) in the set \( S \). The state \( s_i \) corresponds to the state \( \langle q, t \rangle \) as the sequence of Eq.(5), where \( q \in \{q_1, \ldots, q_n\} \) and \( t \in \{\text{Test}, \text{Normal}\} \). The belief state \( b \) can be seen as a vector, and one element \( b_i \) denotes the probability of a state \( s_i \) according to the sequence of all states in Eq.(5).

\[
[s_1, \ldots, s_i, \ldots, s_{2n}] = [\langle q_1, \text{Test} \rangle, \ldots, \langle q_n, \text{Test} \rangle, \langle q_1, \text{Normal} \rangle, \ldots, \langle q_n, \text{Normal} \rangle]
\]  

(5)

Example 3.3. For the states in Example 3.1, the sequence of the states can be denoted as \([<0.2, \text{Test}>, <0.4, \text{Test}>, <0.6, \text{Test}>, <0.8, \text{Test}>, <0.2, \text{Normal}>, <0.4, \text{Normal}>, <0.6, \text{Normal}>, <0.8, \text{Normal}>] \).

Using a belief state \( b \), the current worker’s accuracy can be calculated via Eq.(6). According to Eq.(6), the \( i = 1 \) to \( n \) is corresponding to \( \langle q_1, \text{Test} \rangle, \ldots, \langle q_n, \text{Test} \rangle \) and the \( i = n + 1 \) to \( 2n \) is corresponding to \( \langle q_1, \text{Normal} \rangle, \ldots, \langle q_n, \text{Normal} \rangle \).

\[
acc = \sum_{i=1}^{n} b_i \cdot q_i + \sum_{i=n+1}^{2n} b_i \cdot q_{i-n}
\]  

(6)

Example 3.4. Under the state sequence in Example 3.3, if the current belief state \( b \) is \([0.25, 0.25, 0.25, 0.25, 0.0, 0.0, 0.0, 0.0, 0.0] \), the worker’s accuracy can be calculated as 0.5.

3.1.4. Observations

When the controller agent takes a test action, the worker will receive a test task, assuming that the worker is not aware of the difference between test tasks and normal tasks. If the controller agent gains an answer from the worker, an observation can be known, such as True or False, according to the test task’s true answer. However, when the controller agent takes an assign action, the worker can receive a normal task, so the controller agent can also get a worker’s answer, but we cannot make an exact observation about right or wrong because of unknown true answer. For making this model running, we assume the system can provide a ‘judgment’ about Satisfactory or Unsatisfactory. The judgment should be as reasonable as possible and there will be more details about how to make a judgment in Section 3.2.
3.1.5. **Observation function**

The probabilities of receiving positive observations (True and Satisfactory) are related to the quality level $q$ of the worker. $O(s', a, o)$ denotes the probability that the agent receives an observation $o$ from state $s'$ after having taken the action $a$, as Eq.(7)-(10).

\[
O(<q, \text{Test}>, \text{test, True}) = q, \ \forall q \in Q \tag{7}
\]

\[
O(<q, \text{Test}>, \text{test, False}) = 1 - q, \ \forall q \in Q \tag{8}
\]

\[
O(<q, \text{Normal}>, \text{assign, Satisfactory}) = q, \ \forall q \in Q \tag{9}
\]

\[
O(<q, \text{Normal}>, \text{assign, Unsatisfactory}) = 1 - q, \ \forall q \in Q \tag{10}
\]

**Example 3.5.** The probabilities of $O(<0.6, \text{Test}>, \text{test, True}) = 0.6$ and $O(<0.8, \text{Normal}>, \text{assign, Unsatisfactory}) = 0.2$. The probability graph for the prior examples of POMDP model can be showed as Figure 3. The red nodes and blue rectangles are respectively states and observation. The solid line denotes the transition probabilities under the action and the dotted line is the probabilities of receiving the observation.

3.1.6. **Rewards**

Here we use $R(s, s', a, o)$ to donate the reward that the state transfers from $s$ to $s'$ when taking an action $a$ and receiving an observation $o$. Specifically, the reward is related to the received observation $o$ after taking action $a$ (performing a test or normal task). There are four observations: True or False for the test task, and Satisfactory or Unsatisfactory for the normal task. The reward function can affect the formation of the policy, because the optimal policy is to maximize the expected reward. Intuitively, when the worker's answer to the normal task receives a Satisfactory observation, the reward should be positive, called $R_{S}$; on the contrary, when receiving an Unsatisfactory observation, the controller agent incurs a penalty (the reward $R_{U}$ is negative). When assigning a test task but receiving a True observation, there is also a little penalty (the reward $R_{T}$ is negative), because it has wasted an opportunity to perform a normal task correctly; for the False observation, the reward $R_{F}$ should be slightly larger than $R_{T}$, because it similarly consumes a test task but escapes to make a mistake for normal tasks. We assume that the target accuracy of task answers is $t_{a}$, thus, if the correct probability of workers is larger than the target accuracy $t_{a}$, the controller agent prefers to assign a normal task to this worker. In other words, the following

![Figure 3. The probability graph for POMDP transitions and observations.](image-url)
inequality needs to be satisfied. In practice, the reward function setting should consider the quality requirements for the task completions (i.e. $t_a$).

$$R_T \cdot t_a + R_F \cdot (1 - t_a) < R_S \cdot t_a + R_U \cdot (1 - t_a)$$

We take an example to illustrate how to decide the reward function as shown in Table 1. If the observation is True, the reward $R_T$ is $-3$, because it wasted a chance to complete a normal task correctly. For the False observation, the reward $R_F$ is 0, because it avoids making a mistake. We assume that the target accuracy of task answers $t_a$ is 0.6 and the reward of Satisfactory observation $R_S$ is 30. According to the above inequality, the reward of Unsatisfactory observation $R_U$ can be $-50$.

### 3.1.7. Initial belief state

Because we cannot determine the accuracy of the worker at the beginning phase, the initial belief state is set as a uniform distribution among different quality levels. To be prudent, we assume that every worker is in Test mode in the first phase. Therefore, the initial belief state for the prior example is $[0.25, 0.25, 0.25, 0.25, 0.0, 0.0, 0.0, 0.0]$.

### 3.1.8. Policy

In MDP, a policy $\pi$ is a mapping from state $S$ to action $A$, specifying an action to be taken in each state (Cassandra, Kaelbling, and Littman 1994). In POMDP, the belief state $b$ is defined as the internal state to denote a probability distribution of all the environmental states. Thus, POMDP’s policy $\pi$ can be a mapping from belief state $b$ to action $a$. According to Bayes theorem, the update of belief state $b$ is related to the last belief state $b'$, last action $a$ and received observation $o$ (Cassandra, Kaelbling, and Littman 1994). For example, if the initial belief state is $[0.25, 0.25, 0.25, 0.0, 0.0, 0.0, 0.0, 0.0]$, after taking a test action and receiving a True observation, the belief state can be updated as $[0.2, 0.2, 0.2, 0.4, 0.0, 0.0, 0.0, 0.0]$. Because the belief state can be iteratively updated, we can utilize the information about the last action and its received observation to make a policy. The solved policy can be constructed as a policy graph, which means first taking an action and then according to the received observation, taking the next action. An optimal policy $\pi^*$ is the policy that has a maximal expected sum of discounted rewards after many steps in probability graph. Because taking an action should not only calculate the current step's reward but also consider the future reward. We use an example to illustrate the policy graph. If the reward function is defined as Table 1, the policy graph under the prior POMDP example's transitions and observations can be denoted as Figure 4, where the discount factor is 0.75.

In the Figure 4, the node denotes the action (test or assign) that the controller agent should take and the start node is a test action node. Each node has two outgoing edges that represent the received observation. According to the received observation, we can get the next action node and take its corresponding action. For example, as shown in the policy diagram in Figure 4, we first take the test action and send a test task to the worker, if the worker’s answer is right and we get the True(T) observation, we can jump to the next left node and take test action; otherwise, we should take the next right node’s action. Then, the process is repeated.

<table>
<thead>
<tr>
<th>Observation</th>
<th>True(T)</th>
<th>False(F)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Test</strong></td>
<td>$-3$</td>
<td>$0$</td>
</tr>
<tr>
<td><strong>Satisfactory</strong></td>
<td>$30$</td>
<td>$-50$</td>
</tr>
<tr>
<td><strong>Normal</strong></td>
<td>$30$</td>
<td>$-50$</td>
</tr>
</tbody>
</table>
3.2. Observation maker

Our proposed POMDP method is a sequential decision-making model which is based on the assumption that observations are related to the worker’s accuracy, besides, its optimal policy delineates how to take an action based on received observations. Therefore, how to give the agent a realistic observation is a critical issue.

When the worker in Test mode, due to the known truth of the test task (assuming full trust), compared with the worker’s answer, the system can give a direct observation about True or False. However, for a normal task, the system is not able to know the true answer, so an observation, Satisfactory or Unsatisfactory, cannot be made through this intuitive approach.

Although for a normal task the true answers cannot be found, the observations can be evaluated based on other information. Thus, we consider how to determine the observation types (Satisfactory or Unsatisfactory) as a prediction problem.

3.2.1. Features

According to the historical record data, we extract proper features to get the training sets and train a binary classifier as an observation maker. Thus, we discuss the feature engineering from three kinds of domains: worker, task and temporal in Table 2.

a) Worker Features. Intuitively, the worker’s current accuracy is an important factor affecting the credibility of his or her answer. If workers need to log into the site or their information can be recorded by the crowdsourcing system, workers’ historical records and profiles are exploited as valuable characteristic data. For instance, user reputation appears as trustworthiness of a worker based on recent and historical data (Kantarci and Mouftah 2014); as mentioned in (Kazai, Kamps, and Milic-Frayling 2012), the worker’s age may result in a considerable impact on the answer accuracy.

b) Task Features. Although the true answer of a normal task cannot be immediately known by the system, other crowd’s current answer information, such as answer distribution and answer number can be an important consideration to reflect the observation classes. Zhang et al. (2016) proposed an approach to infer the ground truth of crowd tasks considering the pattern of label
class distribution. Furthermore, the inherent *attributes* of the task can also be viewed as a reference of the observation type. For example, the length of the description text may reflect the difficulty or clarity of the task.

c) *Temporal Features.* The continuity of a worker’s state in temporal dimension is also a factor worth considering, so we use the number of test or normal tasks before the current task to exhibit this property. Besides, the time that a worker spends for the task is likely to affect the quality of the completed task (Kazai, Kamps, and Milic-Frayling 2012).

### 3.2.2. Random Forests Classifier

As the true answer to the normal task is unknown, we utilize the classifier to predict the observation of normal tasks. Owing to the outstanding performance and ease in building for practical applications, Random Forests is adopted as the classifier in our observation maker. Random Forests (RF) is a popular ensemble learning method that was developed by Ho (1995) and Breiman (2001). Classical Random Forests classifier is a collection of decision tree predictors combined via a bagging way. Each un-pruned tree is trained independently by using a randomized sample of the training data with replacement. The tree node are calculated by using the best split among the $k$ subset of all features. Therefore, there are mainly two parameters in Random Forests: the number of trees and the number of selected features at each node. The random process makes the model stronger, more robust and less likely to be over-fitting for the training set. After a number of trees is generated, they obtain the final class by majority voting. Thus, another benefit of Random Forests is the ability to provide the confidence level $P$ for the predicted target, according to the votes of the collection of decision trees.

According to the prediction results (*Satisfactory* or *Unsatisfactory*) of the classifier, our observation maker can directly report the observation. However, the intuitive decision rule does not consider that for different prediction results the losses caused by wrong predictions are different. For example, the loss caused by deciding *observation* Satisfactory when the true observation is Unsatisfactory, is different from the vice versa situation. So, we should reconsider a decision rules according to the different losses caused by different prediction errors.

For notational simplicity, let the predicted results, *Satisfactory* and *Unsatisfactory*, denoted by $s$ and $u$ respectively. The probabilities of the predicted targets for $s$ and $u$ are respectively denoted as $P_s$ and $P_u$. In addition, $\lambda_{ij}$ is defined as the loss caused by deciding $j$ when the true observation type is $i$. Let risk $R_i$ be an expected loss of a given observation type $i$ (Duda, Hart, and Stork 2001), so $R_s$ and $R_u$ can be calculated as Eq.(11)(12).

$$R_s = \lambda_{s,s} \cdot P_s + \lambda_{u,s} \cdot P_u$$

$$R_u = \lambda_{s,u} \cdot P_s + \lambda_{u,u} \cdot P_u$$

Our decision rule is to minimize the expected risk and determine the target observation ($s$ or $u$) with smaller risk. In other words, we only decide $s$ when $R_s < R_u$. According to Eq.(11)(12) and $R_s < R_u$, we can obtain Eq.(13), assuming $\lambda_{s,u} - \lambda_{s,s} > 0$.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Answer</td>
<td>Worker</td>
<td>Worker’s answer for the task</td>
</tr>
<tr>
<td>2) Accuracy</td>
<td>Worker</td>
<td>Worker’s current accuracy for tasks</td>
</tr>
<tr>
<td>3) Historical Record$^2$</td>
<td>Task</td>
<td>Worker’s history information, e.g. reputation, rank</td>
</tr>
<tr>
<td>4) Profile$^2$</td>
<td>Task</td>
<td>Worker’s personal information, e.g. education, age</td>
</tr>
<tr>
<td>5) Answer Distribution</td>
<td>Task</td>
<td>Task’s current answer distribution</td>
</tr>
<tr>
<td>6) Answer Number</td>
<td>Task</td>
<td>Task’s current number of all answers</td>
</tr>
<tr>
<td>7) Attribution$^2$</td>
<td>Task</td>
<td>Task’s own characteristic, e.g. text length, image</td>
</tr>
<tr>
<td>8) Test Task Number</td>
<td>Temporal</td>
<td>Number of test tasks before the current task</td>
</tr>
<tr>
<td>9) Normal Task Number</td>
<td>Temporal</td>
<td>Number of normal tasks before the current task</td>
</tr>
<tr>
<td>10) Time$^2$</td>
<td>Time$^2$</td>
<td>Time it takes the worker to fulfill the task</td>
</tr>
</tbody>
</table>
Because the loss of making a proper decision, $\lambda_{u,u}$ and $\lambda_{s,s}$, can be seen as zero, we can only consider the setting of $\lambda_{u,s}$ and $\lambda_{s,u}$. In practice, the loss of mistake for making a Satisfactory observation for an unreliable (unsatisfactory) answer $\lambda_{u,s}$ is higher than that of making an Unsatisfactory observation for a reliable (satisfactory) answer $\lambda_{s,u}$. Therefore, the higher loss ratio threshold $h_r$ is beneficial for keeping prudent observations and further improving the quality of task completions, which will also be reflected in the experiment section.

3.3. Rejection strategies

In volunteer crowdsourcing, workers freely participate in the completion of tasks, so they cannot be arbitrarily fired except some special situations. However, to maintain the robustness and integrity of the system, the reject option is indispensable for ensuring the quality of task completions. Our rejection strategies are considered from two domains: workers and answers.

a) Reject Malicious Worker. For the binary option tasks, if the worker does not have any prior knowledge, and guesses the answer (assuming that the true answer satisfies a uniform distribution), the correct answer likelihood should be half and half, with approximately 0.5 probability. Therefore, when a worker’s accuracy is not over 0.5, the worker can provide valueless or even harmful information. Our POMDP model can estimate the worker’s accuracy $acc$, as Eq.(6), which can be used to reject workers with $acc < 0.5$. More generally, the threshold to discern malicious workers is donated as $r_w$.

b) Reject Dubious Answer. As we know, prediction confidence of the classifier is an important consideration. Similarly, if the observation maker cannot have enough confidence about its predicted observation type, we need to avoid the risk by rejecting the dubious answer. We define a threshold $r_a$ to ignore the answer when the observation maker’s confidence $P$ is lower than it. If the answer is dubious and rejected, the relevant task still can be assigned to other workers in our algorithm. In the process to judge whether the worker’s answer is dubious, we need to consider the current worker’s performance and task’s information. Therefore, the worker’s answer is seen as dubious does not affect the worker’s other answers or other workers’ answers to the task. The worker can perform other tasks and the task can still be assigned to other workers.

3.4. Algorithm

This section will introduce how the overall algorithm runs in a volunteer crowdsourcing system. The POMDP model is solved offline and the optimal policy can be constructed as a policy graph. As shown in Figure 4, the node of the policy graph is an action (test or assign) and the edge is denoted as an observation (True, False, Satisfactory or Unsatisfactory). The start node of the policy graph is denoted as Node_s, and Node_s.action represents the action of this node. The predictor in the observation maker can also be trained offline by other task data and used to predict observations in Normal mode.

For each worker, there is a controller agent who decides the sequence of test tasks and normal tasks according to the observations. Because the parameters in POMDP models is certain, POMDP should only be solved once. In addition, the controller agent repeatedly calls the optimal policy graph to decide when to allocate a test or normal task at each time step. And the belief state is updated according last belief state, last action and current observation via a straightforward application of Bayes’ rule (Cassandra, Kaelbling, and Littman 1994). In addition, rejection strategies for workers or answers are embedded in the procedure. The whole process is described in Algorithm 1.

The algorithm requires a policy graph which can be calculated in advance, and the function Decision input also contains the current node CurNode (initially root, Node_s) and the belief state BeliefState. This function is executed by each controller agent repeatedly, and the inputs are updated by last execution results. The first line sets the current Action by CurNode’s action. According to
current action, the worker is assigned a task (line 7–14 for test tasks and line 15–23 for normal tasks). The test task or normal task is randomly selected for the worker (line 8 and 16). The variable Answer is represented as the given answer from the worker about a test task (line 9) or a normal task (line 17).

In the Test mode, the observation is given according to the truth of the task (line 10–14). And after assigning a normal task, the system can make an observation (line 19–23) according to the observation maker (line 18). The variables $O$ and $P$ in line 18 are respectively represented by the observation ($Satisfactory$ or $Unsatisfactory$) and the prediction confidence. No matter what the observation is, the answer of the worker except in doubt is accepted by the system and accumulated for further calculating answer distribution score for other workers. After completing the process above, based on the observation (variable Observation), current node (variable CurNode) and policy graph, the next node is known, which is used to reset the variable CurNode (line 24), and the current belief state BeliefState is also updated (line 25). Finally this function returns results that are used for the next time execution.

As Eq.(6), the worker’s current accuracy can be calculated in line 3, which is used to discern malicious workers. If the accuracy is lower than the threshold $r_w$ (line 4–5), the worker is deemed to be malicious and ignored by the system. Similarly, for normal tasks, the prediction confidence $P$ (line 18) of the observation maker is considered as a criterion to reject the answers. If $P$ is no more than the threshold $r_a$, the dubious answer is ignored (line 28–29), but the worker can continue to perform other tasks. We judge whether the worker’s answer is dubious based on the worker’s current performance and the task’s information (see the features in Section 3.2.1). Thus, the dubious answer cannot affect the worker’s other answers or other workers’ answers to the task. The task can still be assigned to other workers and the worker can continue to perform other tasks.

Algorithm 1: Decision($CurNode$, $BeliefState$, $policyGraph$)

```plaintext
Input: Current node $CurNode$ in policy graph $policyGraph$, and belief state $BeliefState$ of POMDP model
Output: Next node, belief state and answer
1   Action ← $CurNode$.action;
2   Observation ← null;
3   $CurAccuracy$ ← $CalCurAccuracy$($BeliefState$);
4   if $CurAccuracy < r_w$ then /* Reject malicious worker */
5       return;
6   end
7   if Action is test then /* Test mode */
8       taskT ← RandomSelectTestTask();
9       Answer ← AssignTestToWorker(taskT);
10      if Answer = taskT.truth then
11          Observation ← True;
12      else
13          Observation ← False;
14      end
15   else if Action is assign then /* Normal mode */
16       taskN ← RandomSelectNormalTask();
17       Answer ← AssignNormalToWorker(taskN);
18       ($O$, $P$) ← ObservationMaker(taskN, Answer);
19       /* Predicted observation $O$ and confidence $P$ */
20       if $O$ is s and $\frac{P}{1-P} > h_r$ then
21          Observation ← Satisfactory;
22       else
23          Observation ← Unsatisfactory;
24       end
25   CurNode ← PolicyGraph($CurNode$, Observation, $policyGraph$);
26   $BeliefState$ ← GetBeliefState($BeliefState$, Observation);
27   if $P > r_a$ then
28       return CurNode, $BeliefState$, Answer; /* Reject dubious answer */
29   else
30       return CurNode, $BeliefState$;
31 end
```
4. Experiments

In this section, we conduct experiments on the real datasets to evaluate the method’s performance, and then compare the results with those of benchmark methods. In addition, we analyze the effects of different parameters on our method.

4.1. Experiment setting

4.1.1. Datasets

There are four real datasets that are adopted by the experiments. The first is the NIST TREC Crowdsourcing Track Task 2 dataset\(^3\) (Buckley, Lease, and Smucker 2010). The original dataset contains 89,624 relevance judgments (1 for relevant, 0 for non-relevant) from 762 workers. This dataset is extracted from the original temporal order of the worker’s relevance judgments. In the TREC(20) dataset, workers who completed less than 20 tasks are excluded and there contains 135 workers. There are 3275 records that have been labeled by expert NIST assessors, thus these records with ground truths are finally included in the experimental dataset. In addition, there are two real datasets (Lin, Daniel, and Weld 2012) that are collected from Amazon Mechanical Turk (AMT),\(^4\), called LinTag and LinWiki. The worker should complete an Entity Linking task in which a sentence and a mention (a portion of the sentence) is shown, and the worker is asked to match the mention to correct Wikipedia entry (Bragg and Weld 2016). Both datasets include 110 questions. A total of 149 workers supplied 3,999 answers in LinTag, and 135 workers supplied 3,857 answers in LinWiki. In the fourth dataset, there are totally 143 workers and 8885 answers. The dataset is collected from Amazon Mechanical Turk on 150 Named Entity Disambiguation questions (NED), which is supplied by Rajpal, Goel, and Mausam (2015). Because these above three datasets (LinTag, LinWiki, Rajpal) do not contain timestamp information, we randomize the order of answers from each worker, as the same setting in Bragg and Weld (2016).

4.1.2. Model settings and environments

In the POMDP model setting, there are 8 states \(s = (q, t)\), where \(q \in Q\) and \(Q = \{0.2, 0.4, 0.6, 0.8\}\). In addition, \(t\) indicates the status of work mode, \(t \in T\), \(T = \{\text{Test}, \text{Normal}\}\). The reward discount factor is uniformly set to 0.9. If the observation is \text{True}, the reward is \(-3\) because it expends a chance to complete the normal task correctly. For the \text{False} observation, the reward is 0, because it can escape to make a mistake. The reward and penalty for the observations, \text{Satisfactory} and \text{Unsatisfactory}, can be seen as parameters of our method and their influences for results will be tested. The loss ratio threshold \(h\) can be considered as the factor that controls the tightness of the observation maker. Furthermore, the effects of the reject options can also be explored in this section.

In the experiments, we first use the \text{appl POMDP} (a fast approximate POMDP tool-kit\(^5\) (Kurniawati, Hsu, and Lee 2008; Ong et al. 2009)) solver to solve our POMDP model offline, where does not exceed 30 seconds (under 2.67GHz CPU and 4GB RAM). Then the solved optimal policy can be constructed as a policy graph (limited with maximum graph depth 6000) that can be repeatedly used. Because the LinTag, LinWiki, and Rajpal datasets do not contain timestamps, the order of answers for each worker is randomized 150 times. The predictor in the observation maker is trained offline based on the sample data from other workers’ and tasks’ records. All experiments are performed on a Windows 7 64-bit PC with 16.0 GB RAM and I7–4770 (3.4GHz) CPU, which are implemented by Python 2.7.

4.1.3. Evaluation criterion

In this paper, we focus on the task’s quality assurance in volunteer crowdsourcing. We adopt the accuracy of task answers to measure the task completion quality. In the process of the normal task assignments, the normal tasks’ true answers are assumed unknown. After all workers’ answers to the normal tasks were acquired, we can compute the accuracy of the completed
normal tasks by comparing with the ground truth. In other words, the accuracy denotes the ratio of correct worker answers in all normal tasks. Specifically, the accuracy is defined as Eq. (14), where $N$ is the number of completed normal tasks and $N_r$ denotes the number of correct normal tasks.

$$\text{Accuracy} = \frac{N_r}{N} \quad (14)$$

This paper mainly focuses on the situation of volunteer crowdsourcing where the number of the free workers is large but the ability level of workers are diverse and variable. Therefore, the task completion quality needs more attention. To some extent, our test mechanism sacrifices some amounts of task completions to ensure the qualities of the task completions. Thus, we also compare the number of completed normal tasks. Since the test mechanism may reject workers, we assume that the workers in the datasets can be reutilized until the total number of completed tasks reaches the original number of tasks completed by workers in the datasets. In addition, for the volunteer crowdsourcing, a practical and effective method is needed. Thus, we also roughly compare the complexity of the decision model, e.g. the runtimes of model training or update frequency of the decision model.

4.1.4. Comparison methods

To evaluate our method’s performance, we implement two baseline methods for comparison. The first baseline method (baseline1) is directly based on no strategy, which means that there is no test task and the allocated tasks to the workers are all normal tasks (with unknown true answers). Another baseline method (baseline2) adopts the basic policy, which first allocates fixed $k$ test tasks for each worker. Only if the accuracy of the test tasks is up to threshold $b$, this worker has the opportunity to continue doing the remaining normal tasks.

We also compare our method with Jung and Lease (2015)’s oracle method and Bragg and Weld (2016)’s method. Jung et al. present a novel prediction model with limited supervision to predict the correctness of workers’ answers. The oracle version of Jung and Lease (2015)’s method assumes all prior tasks’ truth are known and predicts the next label, which is best situation of this kind of prediction model (Jung and Lease 2015). Bragg and Weld (2016)’s method is also based on the POMDP models, but relies on reinforcement learning to learn the parameters of the POMDP models. Although the POMDP model can be solved within at most 1 minute, the system should recalculate the POMDP policy based on the new estimated parameters several times (Bragg and Weld 2016).

4.2. Experimental results

4.2.1. Tests on the performances of methods
The performance of the different methods under the four real datasets are show in Table 3. First and foremost, we compare the performance in the accuracy of task answers (Accuracy) that reflects the task completion quality, as defined in Section 4.1.3. All methods that we implemented were repeated 30 times to obtain the average results. Our test mechanism is under the parameter condition (Reward = 30 for Satisfactory observation, Penalty = -90 for Unsatisfactory observation, $h = 3.0$, and $r_w = r_a = 0.65$). As shown in Table 3, our method outperforms the other five methods under four different real datasets. Compared to the TREC dataset, the LinTag, LinWiki and Rajpal datasets do not contain the information about the sequence of task execution, and then we randomize the order of tasks, so our approach’s superiority in the accuracy of task completions is little less.

In addition, we compare the number of completed normal tasks (with unknown true answers). From Table 3, comparing with baseline1 and baseline2($k = 7, b = 0.85$), our test mechanism sacrifices some amounts of task completions, which is used to test the worker’s current performance and further improve the accuracy of task completions, thereby ensuring the qualities of the
tasks. However, our method can complete more normal tasks than the Bragg’s method, even more than baseline2(k = 15, b = 0.8) in some cases.

We also roughly compare the complexity of the decision model of different methods. The Bragg’s method needs to estimate 5 parameters by reinforcement learning, and then the optimal policy of the POMDP-based decision model should be recomputed according to the new estimate parameters. The solve time of the POMDP model is within 1 minute but the process should be repeated 10 times. In our test mechanism, because the parameter in POMDP need not to be estimated, the decision model should only be computed once, and it takes less than 30 seconds (under one 3.4GHz CPU). The Jung’s method should train a classifier for each worker and the classifier should be updated over time according the worker’s answers to test tasks. In the observation maker of our test mechanism, the classifier that predicts the observation type should only be trained once.

4.2.2. Tests on the effects of pomdp’s reward/penalty setting
We conduct experiments to test the effects of POMDP’s Reward/Penalty setting. In the experiments of Figure 5, our test mechanism is under the parameter condition (hr = 3.0, rw = rw = 0.65). As shown in Figure 5, we set the Reward/Penalty for the Satisfactory and Unsatisfactory observation in POMDP model as 30/-90, 25/-95 and 25/-110 respectively, and then under four different datasets, we compare the accuracy of answers (Accuracy) and the number of completed tasks (Task Number). From Figure 5, it can be seen that the accuracy improves with the increasing of Reward/Penalty ratio, because the higher Reward/Penalty ratio in the POMDP model can lead to a more prudent strategy to decrease the penalty incurred by unsatisfactory task answers. However, the improvement of the accuracy requires a reduction in the number of completed tasks.

4.2.3. Tests on the effects of loss ratio threshold
Although the loss ratio threshold hr cannot directly influence the correctness of tasks’ answers, it can influence generation of observation (Satisfactory or Unsatisfactory) from the observation maker. If the threshold is high, it is rigorous to let the observation maker provide a satisfactory evaluation. Unsatisfactory observations may turn a worker from Normal mode to Test mode to ensure the
worker’s reliability through performing test tasks. In the experiments of Figure 6, our test mechanism is under the parameter condition (Reward = 30 for Satisfactory observation, Penalty = -90 for Unsatisfactory observation, \( r_w = r_a = 0.65 \)). As show in Figure 6, the higher loss ratio threshold \( h_r \) shows the improvement of accuracy, although the progress is relatively smaller than that of the increasing of Reward/Penalty ratio in Figure 5.

4.2.4. Tests on the effects of the features in the observation maker’s classifier
In the observation maker, we utilize the classifier to predict the observation type of a normal task’s answer (Satisfactory or Unsatisfactory). The feature selection of the classifier is defined as Table 2. Because the four real datasets only provide the (1)(2)(5)(6)(8)(9) feature, these features are adopted as the original features. Then, we select the subset of the original features, (1)(2)(5)(8), called subset features. In the experiments of Figure 7, we test the effects of the two feature sets (original features and subset features) on the accuracy of answers (Accuracy) and the number of completed tasks (Task Number). Our test mechanism is under the parameter condition (Reward = 30 for Satisfactory observation, Penalty = -90 for Unsatisfactory observation, \( h_r = 3.0, r_w = r_a = 0.65 \)). As shown in Figure 7, the reduction in the classifier’s features can lead to a decrease in the accuracy of the task answers; moreover, the subset features also diminish the number of completed tasks under three real datasets. The subset features have an influence on the accuracy of the classifier, thereby leading to inaccurate prediction of observation types, which can affect the detection of worker states and the assignment of normal tasks.

4.2.5. Tests on the effects of the thresholds in rejection strategies
Rejection module is a significant part of our test mechanism. We conduct a series of experiments under different datasets to evaluate how the rejection strategies affect the method’s performance.
The factor $r_w$ and $r_a$ respectively denote the threshold to ignore malicious workers and dubious answers. Our test mechanism is under the parameter condition (Reward = 30 for Satisfactory observation, Penalty = -90 for Unsatisfactory observation, $h_r = 3.0$).

As shown in Figure 8, with the increase of the threshold $r_w$, the accuracy of task answers (Accuracy) and the number of completed tasks (Task Number) gradually increase, because if a worker’s accuracy is lower than $r_w$, the worker will be rejected, and the test task assignment will also terminate, further a larger number of normal tasks can be completed by other high accuracy workers.

In Figure 9, if we increase the threshold $r_a$, the accuracy of task completions will grow up, but the number of completed tasks will drop down. The reason is because with the increase of the threshold $r_a$, the rejection strategy can ignore a larger number of dubious answers. However, the
Figure 8. Experimental results under different thresholds $r_w$ to reject malicious workers.

Figure 9. Experimental results under different thresholds $r_a$ to reject dubious answers.
answers that was rejected will lead to a reduction in the number of task completions. Additionally, comparing with the threshold $r_w$, the threshold $r_a$ can bring a greater increase in the accuracy of task answers but the task completions are wasted.

### 4.2.6. Case study

In the prior experiments, we have evaluated the method’s performance in the accuracy of task answers. Through the case study, we want to illustrate the effect of our test mechanism from the perspective of the workers. Thus, from the TREC dataset under the setting Reward/Penalty = 30/-90, $h_r = 3.0$, and $r_w = r_a = 0.5$, we select two representative workers (one is tired, the other is diligent) and observe carefully the decisions (test or normal tasks) made by our method, the changes in worker performances (real-time accuracy), and the correctness of worker answers. Figure 10(a) is a worker who is gradually tired. At the initial phase, the worker gradually comes into a good state, so the controller agent assigns some normal tasks. However, when the observation from the maker becomes unsatisfactory, to be cautious, the worker is converted into test mode. Figure 10(b) is a diligent worker. At the beginning the worker occasionally makes one mistake, but the test task indicates that the worker is still in a good state, therefore normal tasks are resumed. From the two examples, we can see that our test mechanism’s decisions are basically meets our intuitive ideas.

### 5. Conclusion and future work

In this paper, we consider the reliability of crowd workers vary over time (up or down), and we determine how to improve the quality of task completions under volunteer crowdsourcing. We adopt a test task insertion approach to detect a worker’s performance dynamically, thereby ensuring that the worker can perform normal tasks (with unknown true answers) when he or she is in reliable states via testing. Even if the worker is currently deemed in a bad state but is not malicious, continuously routing test tasks to this worker, is beneficial for re-assigning normal tasks to this worker in future good performance. We design a decision method based on POMDP model to decide when to assign test or normal tasks. In addition, this model, without complicated parameter estimation, can be more practical to the real world. And the observation maker is proposed to predict the observation types for the operation of POMDP model. Furthermore, the rejection strategies for malicious workers and dubious answers are embedded in the procedure. In this paper, we focus on the binary option task and each task only has one correct answer. Therefore, we adopt the accuracy of normal task answers to measure the task completion quality. Experimental results show that, our method outperforms benchmark methods in the accuracy of task answers. Moreover, the effect of different parameters on the method have also been analyzed.
In this paper, we only consider the volunteer crowdsourcing situation, where the quality of task completions is the major problem and the test task cost is not focused on due to free workers. In the future, we will consider how to improve the accuracy of task answers within a budget while workers with varying reliabilities have different rewards. For a more general situation, the heterogeneity of tasks should also be studied, for example, different worth for task requesters and diverse difficulties of tasks.

Notes
2. Some scenes may not provide this information.
3. TREC Crowdsourcing Track, https://sites.google.com/site/treccrowd/.

Acknowledgments
This work was supported by the National Natural Science Foundation of China (61472079, 61170164, and 61472089), the Natural Science Foundation of Jiangsu Province of China (BK20171363), the Joint Fund of the National Natural Science Foundation of China and Guangdong Province (U1501254), the Science and Technology Planning Project of Guangdong Province (2015B010131015 and 2015B010108006), the Natural Science Foundation of Guangdong Province (2014A030308008), Guangdong Regular University International and Hong Kong, Macao and Taiwan Cooperative Innovation Platform and International Cooperation Major Projects (2015KGJHZ023), the Natural Science Foundation of China Hunan Province (No. 2016J2057), and the Science Foundation of China Hunan Provincial Education Department (No. 15C0546). We thank Jonathan Bragg’s helpful reply for my questions, and we also thank Christopher Lin’s real datasets, LinTag and LinWiki and Matthew Lease et al.’s real dataset, TREC 2011.

Disclosure statement
No potential conflict of interest was reported by the authors.

Funding
This work was supported by the National Natural Science Foundation of China [61170164,61472079,61472089];Natural Science Foundation of Guangdong Province [2014A030308008];Science and Technology Planning Project of Guangdong Province [2015B010131006,2015B010131015];National Science Foundation of China Hunan Province [No. 2016J2057];Natural Science Foundation of Jiangsu Province of China [BK20171363];Science Foundation of China Hunan Provincial Education Department [No. 15C0546];Joint Fund of the National Natural Science Foundation of China and Guangdong Province [U1501254];Guangdong Regular University International and Hong Kong, Macao and Taiwan Cooperative Innovation Platform and International Cooperation Major Projects [2015KGJHZ023];

References


