Quality Control and Cost Management in Crowdsourcing

by

Yue Wang

M.Sc., Simon Fraser University, Canada, 2015
B.Sc., Simon Fraser University, Canada, 2013

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy

in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Summer 2021

Declaration of Committee

Name:	**Yue Wang**
Degree:	**Doctor of Philosophy (Computing Science)**
Title:	**Quality Control and Cost Management in Crowdsourcing**
Committee:	**Chair:** Martin Ester Professor, Computing Science
	Ke Wang Supervisor Professor, Computing Science
	Jiannan Wang Committee Member Associate Professor, Computing Science
	Tianzheng Wang Examiner Assistant Professor, Computing Science
	Jing Gao Examiner Associate Professor School of Electrical and Computer Engineering Purdue University

Abstract

By harvesting online workers' knowledge, crowdsourcing has become an efficient and cost-effective way to obtain a large amount of labeled data for solving human intelligent tasks (HITs), such as entity resolution and sentiment analysis. Due to the open nature of crowdsourcing, online workers with different knowledge backgrounds may provide conflicting labels to tasks. Therefore, it is a common practice to perform a pre-determined number of assignments, either per task or for all tasks, and aggregate collected labels to infer the true label of tasks. This model could suffer from poor accuracy in case of under-budget or a waste of resource in case of over-budget. In addition, as worker labels are usually aggregated in a voting manner, crowdsourcing systems are vulnerable to strategic Sybil attack, where the attacker may manipulate several robot Sybil workers to share randomized labels for outvoting independent workers and apply various strategies to evade Sybil detection. In this thesis, we are specifically interested in providing a guaranteed aggregation accuracy with minimum worker cost and defending against strategic Sybil attack.

In our first work, we assume that workers are independent and honest. By enforcing a specified accuracy threshold on aggregated labels and minimizing the worker cost under this requirement, we formulate the dual requirements for quality and cost as a Guaranteed Accuracy Problem (GAP) and present an efficient task assignment algorithm for solving the problem.

In our second work, we assume that strategic Sybil attackers may coordinate Sybil workers to obtain rewards without honestly labeling tasks and apply different strategies to evade detection. By camouflaging golden tasks (i.e., tasks with known true labels) from the attacker and suppressing the impact of Sybil workers and low-quality independent workers, we extend the principled truth discovery to defend against strategic Sybil attack in crowdsorucing.

For both works, we conduct comprehensive empirical evaluations on real and synthetic datasets to demonstrate the effectiveness and efficiency of our methods.

Keywords: Crowdsourcing; Quality Control; Cost Management; Sybil Attack

Dedication

Dedicated to my parents and my lover.

Acknowledgements

I wish to express my sincere gratitude to my senior supervisor Dr. Ke Wang, who provided great guidance and support to me on both academic research and life during my Ph.D. studies. I would also like to thank my supervisor Dr. Jiannan Wang for his insightful feedback and suggestions on improving the quality of my thesis, as well as Dr. Martin Ester, Dr. Tianzheng Wang and Dr. Jing Gao for spending their valuable time on serving as the chair and examiners for my Ph.D. thesis defence.

I am also very grateful to all my collaborators, lab mates, colleagues and friends around me for their kind help throughout my study, career and life so far.

Finally, I would like to express my special thanks to my parents and my lover, for their continuous and unconditional supports, encouragement and love.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The concept of crowdsourcing was first brought up by Jeff Howe and Mark Robinson in 2006 as a whole new paradigm for outsourcing tasks to the crowd by means of an open call via the Internet. It allows enterprises to quickly benefit from crowd engagement at a low cost by aggregating the knowledge and ideas from the general public. Taking the mobile crowdsourcing for example, we may collect the traffic/accident (or any other event) data at a chosen location from the flow of passing drivers. Compared to the alternative of outsourcing the task to a dedicated third party, crowdscouring could mobilize the entire driver pool and offer an anytime/anywhere solution. Recently, the phenomenon of crowdsourcing grows at a more surprising speed. According to a report released in Business Insider [1], global crowdsourcing market is expected to reach about 155 billion US dollars by 2027, which was already valued over 9.5 billion US dollars in 2018. In this thesis, we will discuss some major issues in crowdsourcing and provide our solutions.

1.1 Background

What is crowdsourcing? Crowdsourcing is the online operation of asking an undefined large group of people to perform human intelligent tasks (HITs) [37]. As shown in Figure 1.1, crowdsourcing allows the client to issue an open call on a public platform so that cheap and dynamic online workers can flexibly contribute at anytime and from anywhere. Such a cost-effective, decentralized and parallel fashion of performing tasks is the main advantage of crowdsourcing over the traditional way of outsourcing, where the client contracts with a specific organizational entity that provide services by its internal employees.

Although the flexibity of crowdsourcing would greatly accelerate the business cycle and encourage creativity, it may also lead to unreliable quality of collected results, due to the diverse knowledge backgrounds of online workers. Also, as the participation of online workers is unpredictable, it is hard to estimate the latency for the completion of tasks.

[1] https://markets.businessinsider.com/

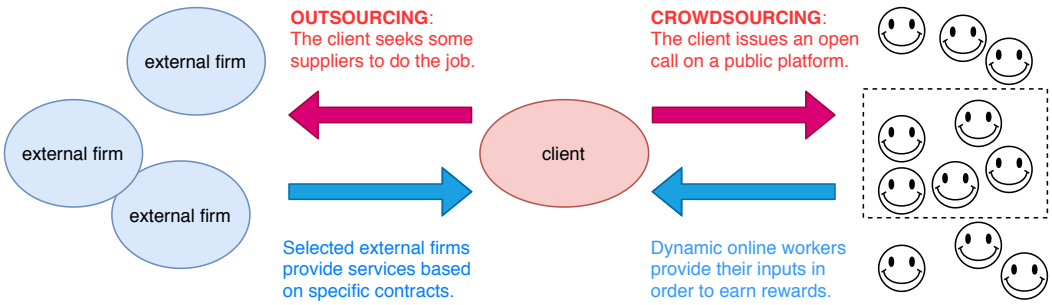

Figure 1.1: Crowdsourcing vs outsourcing

1.1.1 Human Intelligent Tasks

As a problem-solving model, crowdsourcing handles human intelligent tasks (HITs) that require a large computational cost for computer programs but are relatively easy for human beings to solve. Based on the way for workers to complete tasks, HITs in crowdsourcing systems could be classified into the following three types:

- **Single-Choice Task.** A single-choice task asks workers to select a single label out of several optional labels. For example, a task may ask workers to select a sentiment ("positive", "neutral", "negative") of a given sentence. Binary task is a special case of single-choice task with only two optional labels T ("true") and F ("false").
- **Multiple-Choice Task.** Multiple-choice task extends single-choice task by allowing workers to select multiple labels for a task, e.g., labeling a movie with "action" and "comedy" tags. However, as addressed in [69, 111], a multiple-choice task can be easily transformed into a set of binary tasks, e.g., taking each optional tag of a movie as a task and asking workers whether or not the movie should be labeled with the tag.
- **Open-Answer Task.** An open-answer task does not provide any optional answer and relies on workers to input their answers (e.g., a numeric value [74, 53], a translation sentence [13, 104] and worker collected data [87, 30]). While the inherent orderings between numeric values can be studied, it is hard to evaluate the worker quality and the aggregation accuracy for tasks that involves translation and data collection.

1.1.2 Procedure of Crowdsourcing

Modern crowdsourcing platforms, such as Amazon Mechanical Turk (AMT) [2], Flickr [3] and Innocentive [4], allow an HIT requester to recruit online workers for completing a group of

[2]https://www.mturk.com/

[3]https://www.flickr.com/

[4]https://www.innocentive.com/

human intelligent tasks. Let $W = \{w\}$ denote the set of online workers and $T = \{t\}$ denote single-choice tasks in the task group. The procedure involves the following four steps:

1. **Task Publication.** The HIT requester creates a task group $T = \{t\}$ on a crowdsourcing platform, with the instructions about how to complete tasks in $T = \{t\}$, the prerequisite for online workers to participate and the budget that would be used to reward workers for their participation. To estimate the quality of online workers, the HIT requester is often required to provide a small set of golden tasks with known true answer, as denoted by $T' = \{t'\}$.

2. **Task Assignment.** In the online setting of crowdsourcing, the active worker set W is dynamic in that workers may join or leave the system at any time. Usually, a worker in $\mathcal{W}$ must first make a request, indicating that he is ready for the next task, then the system chooses one or more uncompleted tasks from T for the requesting worker. A task is *completed* once no more workers are needed for the task. For simplicity, we assume the task assignment to be non-preemptive, as applied in other crowdsourcing methods, e.g., *iCrowd* [26], meaning a worker must provide an answer for an assigned task before requesting for the next one.

3. **Answer Aggregation.** The answers collected from online workers are represented by $\mathcal{L} = \{l_{t,w}\}$, where $l_{t,w} = null$ if worker w does not provide an answer on task t. As workers with diverse knowledge background may provide conflicting answers to a task, collected worker answers are often aggregated in a voting manner, such as majority voting [102], weighted voting [36] and Bayesian voting [111]. The purpose is to infer the unknown true answer of tasks, as denoted by $\mathcal{L}^* = \{l_t^*\}$, using the aggregated answers of tasks, as denoted by $\mathcal{L}^a = \{l_t^a\}$.

4. **Reward Distribution.** With no ground truth for evaluating worker performance, online workers could be rewarded based on the aggregation result, i.e., a worker would have more award (e.g., reputation or money) if more of his answers agree with the aggregated ones, or simply the number of provided answers. Some works, e.g., FairPlay [28], may allow online workers to set up their expected payment based on their expertise. In this thesis, we assume each worker would be equally paid for each provided answer.

As the application of multiple-choice tasks and open-answer tasks is relatively limited, this thesis focuses on single-choice tasks that have been widely studied in many crowdsourcing works [19, 61, 53, 85, 26, 62, 65], where each worker assigned to a task needs to choose one label from several optional labels. The frequently used symbols are summarized in Table 1.1.

Table 1.1: Frequently used notations

Symbol	Description
$T = \{t\}$	single-choice tasks
$W = \{w\}$	online workers
$T' = \{t'\}$	golden tasks with known true label
$\boldsymbol{\mathcal{L}} = \{l_{t,w}\}$	labels provided by workers on tasks
$\mathcal{L}^* = \{l_t^*\}$	true label of tasks in T
$\mathcal{L}^a = \{l_t^a\}$	aggregated label of tasks in T

1.1.3 Evaluation Measures

Despite the flexible workforce and tremendous profit potential, crowdsourcing faces many challenges due to its open nature. On one hand, workers with unknown quality may provide conflicting labels to the same task, so it is hard to guarantee the correctness of the aggregation results. On the other hand, the cost on hiring online workers could easily become unaffordable with a large size of task group T, even if a much smaller amount of money is paid for each collected worker label, as compared to the traditional outsourcing. By applying various methodologies for task assignment and label aggregation (the second and third steps mentioned above), most crowdsourcing works aim to achieve two goals: (1) maintaining a high aggregation accuracy, i.e., the percentage of tasks whose aggregated label is identical to the true label, and (2) minimizing the cost on hiring online workers. We refer to these two goals as *quality control* and *cost management*.

Let $\mathbb{1}(x, y)$ be an indicator function such that $\mathbb{1}(x, y) = 1$ if $x = y$ and $\mathbb{1}(x, y) = 0$ if $x \neq y$. We formally quantify the measurement of quality control and cost management by *aggregation accuracy* and *worker cost*, respectively.

$$Aggregation\ Accuracy = \frac{\sum_{t \in \mathcal{T}} \mathbb{1}(l_t^a, l_t^*)}{|T|} \tag{1.1}$$

$$Worker\ Cost = \frac{\sum_{l_{t,w} \in \mathcal{L}} (1 - \mathbb{1}(l_{t,w}, null))}{|T|} \tag{1.2}$$

Note that the true labels $\mathcal{L}^*$ are normally unavailable in practice, so the aggregation accuracy would be estimated by conducting experiments on tasks with known true label. Also, the worker cost is measured by the number of workers hired for a task on average because the HIT requester usually pays the same amount of money for each label provided by workers within a task group.

1.2 Research Topics and Contributions

In this thesis, we focus on two research topics raised by the goals of quality control and cost management, which have become the major research directions in crowdsourcing for

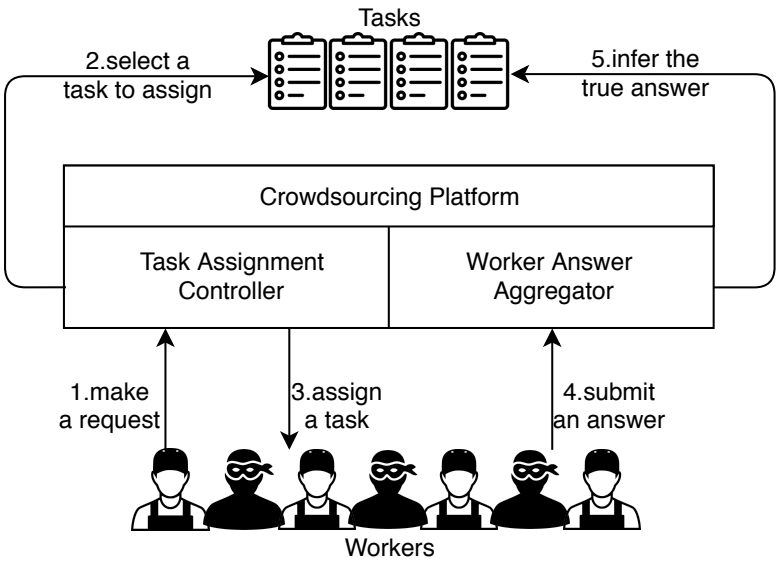

Figure 1.2: The entities in a typical crowdsourcing system

academic and industrial communities. The first topic falls into guaranteeing an expected aggregation accuracy with minimized worker cost, while the second topic focuses on the security issue about defending against Sybil attack in crowdsourcing.

1.2.1 Research topic about accuracy guarantee with cost minimization

As being confirmed by some early studies [8, 80, 81], asking several non-expert online workers to perform a task and aggregating their responses could generate a similar result as hiring experts. Usually, a crowdsourcing method conducts quality control and cost management through two components – *task assignment controller* and *worker label aggregator*, as shown in Figure 1.2. The former one determines the set of online workers assigned to each task, while the later one infers the unknown true label of each task from collected worker labels. Note that for some specific crowdsourcing applications, such as Yelp and MovieLens, the task assignment controller may be omitted because online workers are allowed to choose the restaurants or movies they want to comment.

In the literature, a common approach for quality control is to pre-determine a fixed number of task assignments, either per task [26, 62] or for all tasks [15, 34, 109], and then perform task assignment and label aggregation based on workers' estimated quality for tasks. However, this approach could not provide a specific guarantee for the aggregation accuracy or minimize the worker cost.

Example 1. *Suppose an HIT requester expects at least 0.9 aggregation accuracy for a task group $\mathcal{T}$, and the worker set $\mathcal{W}$ contains seven workers $\{w1, w2, w3, w4, w5, w6, w7\}$, where each worker has 0.75 probability to provide the true label of a task. If each task is assigned to only three workers and majority voting is used for label aggregation, the expected*

aggregation accuracy would be 0.84 according to Poisson binomial distribution [14], which means the expectation of the HIT requester cannot be satisfied. If all the seven workers are hired for each task, the aggregation accuracy would pass 0.9, but the worker cost is not minimized because any five of these workers are good enough for meeting the expected aggregation accuracy. If each worker has at least 0.9 probability to provide the true label of a task, then the minimum worker cost per task should be 1.

As indicated in this example, heuristically pre-determined worker cost may lead to a waste of money or provide no guarantee for the aggregation accuracy. More importantly, the unknown aggregation errors would propagate to downstream applications of crowdsourcing, e.g., learning a prediction model based on the aggregation result. In most cases, allowing a large unknown error in training data is not acceptable, especially for high stake applications in the medical and financial domains, e.g., a system for real-time patient assessment [7] which uses mobile electronic triaging accomplished via crowdsourced information.

To solve this problem, we consider a scenario that the HIT requester expects a threshold for the aggregation accuracy to be satisfied with minimum worker cost. A natural consequence of guaranteeing the aggregation accuracy is that some tasks may not be completed if there is a lack of high-quality online workers in the system. Therefore, instead of completing all tasks without any accuracy guarantee, we want to complete as many tasks as possible with the guaranteed aggregation accuracy and minimum worker cost. Providing the information on the quality of such tasks is important for high stake downstream applications. Note that the remaining tasks could be completed later using a traditional algorithm, but importantly, the HIT requester is informed of whether a task is completed with the accuracy guarantee or not.

Given the above motivation, we formulate the dual requirements for quality control and cost management with the online setting of crowdsourcing as a *Guaranteed Accuracy Problem (GAP)* in Chapter 3. Parameterized by a quality threshold δ and a cost threshold λ, GAP requires its solution to meet three requirements: (i) the probability to infer the true label of a task should be at least δ; (ii) the worker number (cost) for each task should be minimized under (i) but not exceeding λ; (iii) the number of tasks that can be completed is as large as possible under (i) and (ii). We present a solution to GAP, called Guaranteed Accuracy Algorithm (GAA), that assigns tasks to candidate workers who are as good as the top workers in minimizing worker cost with the δ threshold satisfied, and present an iterative GAA ($IGAA$) that completes additional tasks under the accuracy guarantee by iteratively relaxing the minimum cost requirement.

We empirically evaluate the proposed GAA and $IGAA$ on a movie ranking dataset from MovieLens[5] and a semantic analysis dataset from Figure Eight[6]. Experimental results confirmed the accuracy guarantee and worker cost minimization of GAA and $IGAA$ in solving GAP, as compared to the state-of-the-art algorithms.

1.2.2 Research topic about quality control under Sybil attack

The voting based label aggregation of crowdsourcing has an underlying assumption that all workers provide labels to tasks independently and honestly, which may fail in the presence of *Sybil attack* [23]. To earn the reward while spending minimum effort, a Sybil attacker could manipulate several Sybil worker accounts to share a randomized label on each task so that the independent workers on the same task could be outvoted. Consequently, the reputation/weight of Sybil worker accounts would be mistakenly increased because their labels agree with the aggregated labels more often. In this case, the randomized labels of Sybil workers would be weighted more in the label aggregation, which definitely decreases the quality of aggregated labels.

In fact, without requiring a distinct bank account for each worker, the bar for creating Sybil worker accounts could be further reduced. For example, ChinaCrowds[7] allows different workers to use the same bank account for receiving momentary rewards, and AMT provides the alternative of redeeming rewards as a gift card, so the attacker can easily create multiple worker accounts for his attacking purposes.

Sybil attack becomes *strategic* and is harder to defend if the attacker attempts to evade detection [68]. For example, online workers are often required to label *golden tasks* with known true label in a cold-start phase as a *qualification test*. A strategic Sybil attacker can first label golden tasks honestly to improve the trustworthiness of Sybil workers but attack later on normal tasks. Even if a *hidden test* is applied to randomly assign workers to golden tasks, the attacker can identify a golden task when more than K Sybil workers are assigned to the same golden task because a normal task is usually assigned to a fixed number K of workers. To hide the coordination among Sybil workers, the attacker may also allow them to occasionally deviate from the label sharing to make Sybil detection more difficult.

To defend against strategic Sybil attack in crowdsourcing, we propose a framework, called *TDSSA* (Truth Discovery against Strategic Sybil Attack), that extends the principled truth discovery to the scenario that several online workers might be Sybil workers controlled by a strategic Sybil attacker, with the strategic assignment and camouflage of golden tasks.

[5]http://grouplens.org/datasets/movielens/

[6]https://data.world/crowdflower/narrativity-in-scientific-pub

[7]http://www.chinacrowds.com/

We conduct experiments on two public real crowdsourcing datasets, named NLP [110] and DOG [112, 102], and synthetic datasets through attack injection recommended in the literature. Under various data characteristics and attack settings, *TDSSA* can achieve a much higher aggregation accuracy than baseline methods.

1.2.3 Thesis Organization

The rest of this thesis is organized as follows. In Chapter 2, we present some background knowledge on crowdsourcing, including the modeling of tasks and workers, online task assignment techniques and adversarial attacks. In Chapter 3, we describe *GAP* in details, including proposed *GAA* and *IGAA* methodology, as well as some experimental results on two real world data sets. In Chapter 4, we study the problem of strategic Sybil attack and propose a defense framework $TDSSA$ with the verification of its effectiveness under different attacking environments. We conclude this thesis in Chapter 5 with a summary of our contributions and some future directions that are worth exploring.

Chapter 2

Related Work

Nowadays, our lives have been filled with data, such as text, music, pictures and videos, and the explosive growth of data does not seem to slow down a bit. As a cost-effective way for collecting data from the general public, crowdsourcing has been widely used to advance researches that require humans to provide the initial data, such as information retrieval [51], machine learning [84], recommendation system [2] and natural language processing [77]. For example, with the help of machine learning, we are able to derive the meaning hidden in all kinds of data by discovering important patterns from observations and experience. Despite many different methods have been proposed for machine learning, such as supervised learning, unsupervised learning and reinforcement learning, the basic idea is to build a model from sample data (i.e., training data) for prediction and decision making.

In contrast to traditional data collection of outsourcing tasks to some employers or contractors, crowdsourcing enables the HIT requester to pay a small amount of money per task assignment and outputs labeled data with good quality when an appropriate aggregation method is applied, where the aggregation results of crowdsourcing could be used in machine learning as the training set and the verification set. However, the use of crowdsourcing has also raised many challenges because the data provided by dynamic online workers with various knowledge backgrounds or even malicious purposes could have a low quality, which would not only result in a waste of money for collecting data but also lead to a failure in the subsequent use of the data. For example, the performance of machine learning applications highly depends on the quality of input data (i.e., training set). In this section, we systematically review the state-of-the-art methods and potential threats related to the quality control and cost management of crowdsourcing.

2.1 Quality Control

Effective quality control is critical to the success of data collection through crowdsourcing. In the literature, many different approaches, such as enhancing the model of tasks and workers, applying golden tasks to estimate worker expertise, adaptively assigning online workers to

tasks and aggregating worker labels based on worker quality, have been proposed to ensure a high quality of the aggregation result.

2.1.1 Model Enhancement

Model enhancement mechanism attempts to improve the aggregation quality by accurately modeling tasks and workers. The fundamental factor is to ensure a good mapping between worker expertise and task requirement, which allows us to perform online task assignment (Section 2.1.3) by selecting workers with highest chance to provide the true label for each task and aggregate worker labels (Section 2.1.4) by assigning appropriate weights to workers.

Task Modeling

Some early works [98, 62] model the task difficulty as a single value, with the assumption that the labels provided by a worker have different quality w.r.t. the various difficulty of tasks. Intuitively, the more difficult a task is, the less chance a worker has to correctly label the task.

GLAD [98] uses a logistic sigmoid function to estimate the probability $Pr(l_{t,w} = l_t^*)$ that the label $l_{t,w}$ given by a worker w on a task t is the true label l_t^*:

$$Pr(l_{t,w} = l_t^*) = \frac{1}{1 + e^{-d_t \cdot p_w}} \tag{2.1}$$

where d_t indicates the difficulty level of task t and p_w represents the quality of worker w. With a probabilistic graphical model (introduced later) that samples d_t and p_w respectively from two prior distribution parameters α and β, the true label of tasks can be inferred simultaneously with d_t and p_w.

FaitCrowd [62] extends GLAD [98] by introducing task bias in the probabilistic graphical model to capture the difficulty of each task. The idea is that the decreasing of bias would lead to less task difficulty and thus a higher probability to provide the true label. By assuming that the difficulty level of most tasks is moderate while only a small portion of tasks are relatively easy or hard, the bias of a task is recommended to be drawn from a Gaussian distribution.

Task difficulty provides a simple representation for the modeling of tasks, but the fact that a task may require knowledge across different topic domains is ignored. To address this issue, recent works [76, 26, 109] connect task content with explicit features or latent topics by modeling each task t as a vector $\Theta_t = \{\theta_{t,d}\}_{d=1:D}$, where D indicates the size of features or topics.

SmartCrowd [76] represents $\theta_{t,d}$ as a quality threshold for solving knowledge-intensive tasks, where collaborative knowledge content could be created through crowdsourcing. For

example, Wikipedia[1] relies on online workers to gradually increase the content and quality of each knowledge piece, where each worker has certain expertise to contribute. SmartCrowd [76] ensures that the additive quality of online workers on each task t would pass the quality threshold $\theta_{t,d}$ for each topic d, but the solutions for knowledge-intensive tasks usually require prior knowledge about online workers, such as the expected wage, expertise and/or acceptance ratio.

Both iCrowd [26] and Docs [109] treat Θ_t as a topic distribution with $\sum_{d=1:D} \theta_{t,d} = 1$, which is modeled based on the text description of tasks. For each topic d, $\theta_{t,d}$ indicates the relevance of task t to the topic, so a worker w is more likely to provide the true label to a task t if the worker has higher expertise on the task's related topics, i.e., topics with larger $\theta_{t,d}$. The main difference between iCrowd [26] and Docs [109] is that the former constructs a task similarity graph (i.e., tasks with similar topic distribution will be topologically close to each other in the graph) using latent topic modeling techniques, while the later relies on external knowledge base to capture explicit topic domains of tasks.

Worker Modeling

The purpose of modeling task difficulty or topic distribution is to better understand and evaluate the quality of online workers on tasks so that the collection and/or aggregation of worker labels can be improved accordingly.

Many works [45, 19, 60, 6] associate each worker w with a single worker accuracy $a_w = Pr(l_{t,w} = l_t^*) \in [0, 1]$ for all tasks, i.e., a_w represents the probability for w to provide the true label of any task t. This approach allows us to integrate worker quality into worker label aggregation by assigning different weights to the labels provided by different online workers. For example, $a_w = 1$ indicates an expert who always label tasks correctly, while $a_w = 0.5$ reveals a spammer who gives random labels for binary classification tasks with two optional labels. GLAD [98] and CRH [54] extend worker quality to a wider range of $(-\infty, +\infty)$, with the same idea that a worker with higher quality has a higher chance to label tasks correctly.

Corresponding to the topic distribution in task modeling, a worker could have diverse expertise on different topics. For example, a worker who is familiar with computer science may provide a high quality label to a task about information technology, but fail to correctly label a task related to geography if the worker is not interested in this field. Based on this observation, many works [96, 62, 109] map the quality of each worker w into the topic domains using a vector $\Psi_w = \{\psi_{w,d}\}_{d=1:D}$, where $\psi_{w,d} \in [0, 1]$ indicates the quality of worker w on topic d.

When tasks in a task group have a fixed optional label set $\mathcal{L}^o = \{\ell | \ell = 1, ..., L\}$, e.g., image annotation, some works [18, 74, 48, 85, 65] use an $L \times L$ confusion matrix, denoted by

[1] https://en.wikipedia.org/wiki/Main_Page

$\Pi_w = [\pi^{\ell'}_{\ell,w}]_{\ell,\ell'=1:L}$, to represent a worker w's quality, where $\pi^{\ell'}_{\ell,w} = Pr(l_{t,w} = \ell' \mid l^*_t = \ell)$ is the probability for worker w to provide label ℓ' on any task t with true label ℓ. For example, suppose $\mathcal{L}^o = \{True, False\}$ and $\Pi_w = \left[\begin{smallmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{smallmatrix}\right]$. Then, 0.9 and 0.8 are the accuracies of worker w on a task t, given the true label is $True$ and $False$ respectively. Compared to modeling worker quality using a single value, confusion matrix captures more information and thus usually leads to a better aggregation accuracy.

2.1.2 Golden Task

The dynamic flow of online workers in crowdsourcing makes it hard to evaluate worker quality, especially when the ground truth is unknown *a priori*. To address this issue, many works [61, 63, 26, 1, 109, 102] rely on the HIT requester to provide a small number of golden tasks with known true label, denoted by T'.

There are two ways to utilize golden tasks − qualification test and hidden test. In a qualification test, workers are asked to first label golden tasks in T' at a cold-start phase before being assigned to any normal task in T. This introduces a potential risk that spammers may carefully label these golden tasks in order to improve their reputation or weight in worker label aggregation. To solve this problem, the assignment of golden tasks can be mixed with that of normal tasks in a hidden test, so workers cannot distinguish golden tasks in T' from normal tasks in T.

Golden tasks make the estimation of worker quality easier, but they usually incur more costs for two reasons: 1. Hiring experts to give the true label of golden tasks could be very expensive. 2. It is a waste to pay workers for their labels on golden tasks with known true label. In addition, how to choose a sufficient amount of golden tasks for revealing the different domain knowledge of workers is another burden leveraged to the HIT requester. iCrowd [26] selects tasks that have the maximum "influence" to other tasks in a similarity graph as golden tasks, while Docs [109] intuitively ensures that all topic domains and the aggregated topic distribution of all tasks will be captured by the selected golden tasks.

2.1.3 Online Task Assignment

Many early works [92, 30, 61, 62, 19] of crowdsourcing do not distinguish the various quality of workers on different tasks during online task assignment, so online workers are assigned to tasks in a random manner.

To improve the quality of collected worker labels on binary classification tasks, AskIt! [10] applies entropy theorem to estimate the uncertainty of the aggregation result and assigns each requesting worker to a fixed number of tasks with highest uncertainty. By associating each worker w with a skill level $a_{t,w}$ on each task, ATA [36] perform task assignment based on a quantity $q_{t,w} = (2a_{t,w} - 1)^2$ evaluated for each pair of task t and worker w. Intuitively, an informative label will be collected if $q_{t,w}$ is close to 1, while the label is considered to be random noise when $q_{t,w}$ is close to 0. QASCA [111] considers different evaluation metrics

(e.g., Accuracy or F-score) for online task assignment, where a fixed number of tasks that can lead to the maximum improvement in aggregation accuracy are selected for each requesting worker. Both iCrowd [26] and Docs [109] map workers and tasks into topic domains and assign a worker to a task if the domains of the task is within the worker's expertise (e.g., the worker has a high probability to provide the true label).

2.1.4 Aggregation Methods

Aggregation method plays a very important role in the quality control of crowdsourcing, where noisy labels provided by online workers are aggregated to infer the unknown true label. As online workers may have diverse knowledge backgrounds, it is important to assign appropriate weights to their labels in the aggregation.

Direct Computation

The simplest aggregation method is Majority Voting (MV) [30, 26] that regards the label voted by most workers as the aggregated label. Since it ignores different levels of worker expertise and task difficulty, the aggregation accuracy is usually be low. Weighted Voting (WV) [36] distinguish the weight assigned to each worker's labels to improve the aggregation accuracy, where the optional label receives highest weight becomes the aggregated label. For tasks that require workers to provide a numeric value [74, 53], e.g., ranking, the aggregation result could be determined by the mean or median value of all collected values for a task.

Iterative Learning

Without using golden tasks, several works [54, 53, 112, 109, 22, 21, 31, 110, 57, 41] apply Expectation-Maximization (EM) algorithm [20] to automatically estimate worker quality and infer the true label of tasks in an iterative manner. Although different mechanisms have been proposed to deal with various scenarios, a general principle of iterative learning is followed: the label provided by high-quality workers should be weighted more in the label aggregation, and the workers who provide labels more accurately (i.e., match the aggregated label more often) should be assigned higher weights. Based on this principle, the aggregation of worker labels and the estimation of worker weights are tightly coupled through the following two steps in each iteration until convergence:

Step 1: Worker Label Aggregation. The aggregated label for each task is determined by a weighted voting, assuming the weight of each worker is fixed.

Step 2: Worker Weight Estimation. The weight of each worker is estimated by comparing the worker's labels with the aggregated ones computed in Step 1.

The differences among various iterative learning methods can be found in the modeling of workers and optimization functions. For exampl, CRH [54] models each worker's quality as a single accuracy, while CATD [53] considers both worker accuracy and confidence in an

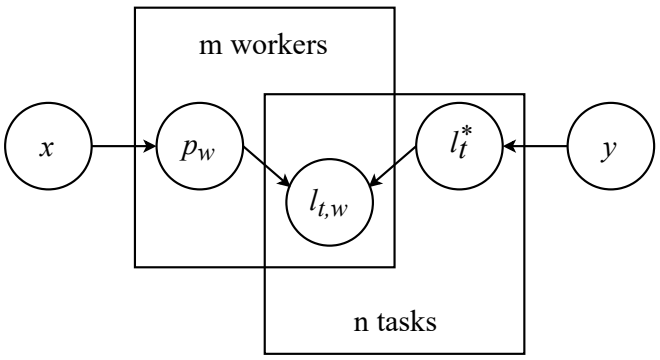

Figure 2.1: A simple probabilistic graphical model (PGM)

objective function such that a worker who often provides the aggregated label (i.e., high accuracy) and labels a plenty of tasks (i.e., high confident) should have a high weight. Minimax [112] associates each worker with a probability distribution that specifies the probability for the worker to provide each optional label and leverages the mini-max entropy theorem into the iterative learning. Docs [109] models each worker as a vector of expertise w.r.t. the topic domains and applies Bayesian voting in Step 1.

Probabilistic Graphical Model

A probabilistic graphical model (PGM) is a graph illustrating the conditional dependency relationships between random variables. Figure 2.1 shows a simple PGM, where each node represents a variable and the two plates contain variables related to m workers and n tasks, respectively. x denotes the prior variable for p_w that represents the quality of workers, while y denotes the prior variable for l_t^* that indicates the true label of tasks. The directed edges model the conditional dependence between two variables, so the child node follows a probabilistic distribution conditioned on the values taken by the parent node. The basic idea of PGM is to infer the unknown variables p_w and l_t^* based on x, y and observed worker labels $l_{t,w}$ through the maximization of the join probability of random variables.

There are many existing works [19, 98, 45, 60, 18, 74, 48, 85, 96, 62] that apply PGM with different modelings of tasks and workers. ZenCrowd [19], GLAD [98], KOS[45] and VI [60] represent each worker as a single quality. DS [18], LFC [74], BCC [48] and CBCC [85] apply a confusion matrix to model the quality of each worker. Multi [96] and FaitCrowd [62] project tasks into topics and capture worker quality over the topic domain. The prior distributions of PGM act as our knowledge or assumption about workers and tasks, which make the presentation easy to understand. However, the introduction of prior distribution and join probability also complicates the inference problem and increase the computational cost. Besides, the inference process normally starts after the collection of worker labels, so the assignment of tasks is usually conducted in a random manner without considering the knowledge requirement of tasks or the various quality of workers.

2.2 Cost Management

There are many different ways to manage the cost on hiring online workers in crowdsourcing, including fixing the number of workers per task, pruning unnecessary tasks, varying the payment for workers and allocating the budget among tasks.

2.2.1 Fixed Assignments

The most widely used cost management [26, 62, 98, 45, 60, 74, 85, 96, 62], is to fix the number of workers per task based on the HIT requester's budget. Such a heuristic decision may lead to a waste of money or inadequate redundancy as the quality of online workers is unknown a priori. For example, if the worker number per task is large while most workers happen to be knowledgeable for providing the true label of tasks, we may hire too many workers for each task. On the contrary, if the worker number per task is small when tasks are relatively difficult for most workers, the aggregation accuracy would be low.

2.2.2 Task Pruning

Many existing works [19, 92, 44, 93, 97, 5, 86, 95, 100] limit the cost of crowdsourcing by removing unnecessary tasks and focusing on the remaining tasks. ZenCrowd [19] proposes a probabilistic reasoning model to identify uncertain entity matching pairs. Crowder [92], GCER[97] and ACD [95] evaluate the similarity between each pair of entities and cluster entities into groups for reducing the number of comparison. LTR [93] and CAER [86] minimize the number of crowdsourced entity pairs by identifying the optimal labeling order from transitive relations. CrowdPlanr [44] and OASSIS [5] use the labels collected for a subset of tasks to infer the true label of other tasks. CrowdGame [100] applies a game-based rule generation method to select candidate rules with large coverage for cost reduction. The main drawback of task pruning is that it only considers the relationship between tasks but ignores the bound between task difficulty and worker quality. Therefore, the reduction of cost cannot be done in a per-task level. In addition, the application of the above works is restricted to certain types of tasks due to the necessity to exploit the domain-specific properties and structures, such as entity resolution and queries with sequential outputs.

2.2.3 Payment Variation

Fairly paying online workers does not only encourage them to make a better contribution, but also attracts more potential workers to participate, as indicated in [29]. Several works [101, 38, 78, 71, 28, 4, 49, 103, 32] have been proposed to match the payment with the quality or expectation of workers in crowdsourcing. FairPlay [28] explores how the fair payment will affect workers' intentions to the labeling of tasks as well as their loyalty towards the crowdsourcing platform. BonusMDP [101] uses worker-centric Markov Decision Process to decide whether and when to place a bonus for improving worker quality or retention.

PostedPrice [38] exploits the features of tasks to derive an optimal payment scheme that requires fewer inputs compared with many existing approaches. By considering from the perspective of online workers, many works [78, 71, 49, 4, 103, 32] also aim to improve online workers' benefits, such as utilities, efficiency and privacy.

2.2.4 Budget Allocation

Cost management can also be conducted through the allocation of the budget to different tasks. SmartCrowd [76] pre-computes an optimal task assignment and adaptively modifies it to handle worker profile updates, but each worker's expertise on different topics and each task's topic-based requirement are assumed to be available before hand. BTSK [34] analyzes the hypernym or hyponym relations among the optional labels of each task to determine which tasks should be assigned in each iteration to gather more worker labels. Docs [109] selects a most beneficial task for a requesting worker based on how much ambiguity in the label aggregation can be reduced if assigning the worker to the task. CDAS [61] and CQTO [1] evaluate a confidence score for each task and terminate hiring more workers for a task once the confidence is high enough to satisfy the HIT requester's expectation on the accuracy of the aggregation result.

2.3 Potential Threats

The main potential threats in crowdsourcing come from dishonest labels, which could be provided by online workers either in a random manner [89], through duplication [22] or from an artificial generator [65]. In addition, the online nature of crowdsourcing makes it untrusted for the privacy issue, where sensitive information of the HIT requester and online workers might be leaked or deduced. With different levels of worker collaboration, attack purpose and knowledge about the system, some major attacks against crowdsourcing systems are listed as follows.

2.3.1 Individual Attack.

An individual attacker is defined as an online worker whose labels significantly deviate from the true label, mainly due to the lack of knowledge for performing the tasks. In order to obtain monetary or reputation rewards as quickly as possible, the attacker typically provides his labels in a random manner [73], i.e., randomly choose one of the optional labels, or alter his own behavior to mimic normal workers [22], e.g., copy and modify other workers' reviews and comments in a recommendation system like Tripadvisor[2] and Yelp[3]. Individual

[2]https://www.tripadvisor.com/Hotels

[3]https://www.yelp.com/

attackers are assumed to have no exact information about the mechanism of the system, e.g., how workers are assigned to tasks and how worker labels are aggregated. Although the redundancy for worker label collection, i.e., assigning several workers to a task, could reduce the influence of an individual attacker, the aggregation accuracy may be low if there are many such attackers in the system.

2.3.2 Group Attack.

Group attack involves the manipulation of several online workers to collaboratively provide a shared label to the same task.

Sybil Attack. Sybil attack [102] in crowdsourcing occurs when an attacker recruit or create a group of online workers to carry out malicious campaigns by sharing the same label on a task. Different from individual copiers, Sybil workers under the centralized control of the attacker would take advantage of voting-based aggregation methods to collaboratively outvote independent normal workers. Since the purpose of the attacker is not to decrease the aggregation accuracy but to dominate the aggregation result for more rewards, the shared labels are usually randomized. Defending Sybil attack is not a new topic in online social networks, but we may not simply adopt the solutions for crowdsourcing because the social structure of dynamic online workers is usually unavailable. A general approach of Sybil defense in crowdsourcing should mainly rely on the analysis of labels provided by online workers.

Data Poisoning Attack. Data poisoning attack aims to inject manipulated training data for corrupt a learning model, such as recommendation system [52], machine learning [40] and crowdsourcing [65]. Usually, a worst-case attacking scenario is assumed for data poisoning, where the attacker may have full knowledge about the system, e.g., the aggregation method and the labels provided by normal workers, so that the error in the aggregation result could be maximized. In a more sophisticated data poisoning attack, the attacker would attempt to hide the identity of malicious workers by agreeing with the majority when the aggregation result is unlikely to be overturned, instead of deviating from the majority on every task.

2.3.3 Privacy Attack.

The online nature of crowdsourcing makes it untrusted for the privacy issue, where sensitive information of the HIT requester and online workers might be leaked or deduced. For example, in spatial crowdsourcing [3], an HIT requester posts a task to the server along with the location for performing the task, and an online worker also needs to disclose his location to the server and travels to the task location after being assigned to the task. Such partial knowledge about workers and tasks could reveal privacy information, such as personal interests, behavior patterns and home/work locations. Privacy issue does not directly influence the quality of aggregated labels but may discourage online workers to participate.

As a special interest in spatial crowdsourcing, protecting the location information related to tasks and workers has become an important research direction. There are three major techniques for privacy protection in spatial crowdsourcing, including spatiotemporal cloaking, differential privacy and encryption.

Spatiotemporal Cloaking

In spatiotemporal cloaking, the locations of tasks and workers are hided inside a cloaked region. By creating an intermediate trusted server that constructs a cloak region for each worker based on locality-sensitive hashing (LSH) and then searches the K-nearest tasks, K-anonymous location privacy is achieved in [88] where each worker's location is indistinguishable from at least $K-1$ other workers. PiRi [47, 46] also guarantees K-anonymous location privacy with a voting mechanism such that neighboring workers select a set of representatives to send out their cloaked region for querying the nearest tasks. In [70], workers are allowed to query tasks by a cloaked region with certain probability distribution, instead of an exact location.

Differential Privacy

Differential privacy [25] is a privacy protection that releases patterns of a group without substantially disclosing any individual. In [82], a sanitized spatial index is constructed by a noisy count of workers at each index node, and the information of a task will be sent to index nodes that contain sufficient workers to complete the task with high probability. PriCSS [42] provides differential location privacy along with the minimization of payment or social cost for spectrum sensing tasks. With worker locations transformed into an obfuscation matrix that encodes the probability of obfuscating any two regions, a data adjustment function is designed in [94] to fit the original sensing data to the obfuscated location.

Encryption

Encryption techniques could be used to transform the locations of tasks and workers into encrypted data and then calculate the distance between two encrypted positions [59, 58]. In [75, 105], workers are asked to provide their travel costs for different tasks, instead of their locations, where the costs will be encrypted into perturbed data and considered for task assignment. A drawback of encryption-based privacy protection is that the computational overhead is very high.

Chapter 3

GAP: Guaranteed Accuracy Problem for Crowdsourcing

The open nature of crowdsourcing makes it hard to deal with the dilemma of quality control and cost management. As online workers with diverse knowledge backgrounds might provide conflicting labels to the same task, the unknown true label of a task is inferred by aggregating labels collected from several workers. With unpredictable quality and arrival of online workers, it is hard to guarantee the aggregation accuracy. This challenge is compounded if we want to also minimize the cost of hiring online workers for each task. In this chapter, we formulate the *Guaranteed Accuracy Problem (GAP)* to capture the duel requirement on guaranteed aggregation accuracy and minimum worker cost, and we present an online algorithm to solve the problem. We evaluate these goals with respect to state-of-the-art methods.

3.1 Background and Overview

By harvesting knowledge and information from random online workers, crowdsourcing provides an efficient problem-solving model for human intelligent tasks (HITs), such as entity resolution [92, 86], sentiment analysis [11] and image classification [67, 61]. In general, the HIT requester hires several online workers for each task (by rewarding them with money or gifts) and aggregates the labels of these workers to infer the unknown true label.

As dynamic online workers may provide noisy and conflicting labels, maintaining a high aggregation accuracy, i.e., the probability for the aggregated label to be identical to the unknown true label, has become the primary goal of crowdsourcing. A common approach is to fix the number of task assignments, either per task [26, 62, 111] or for all tasks [15, 34, 109], and leverage domain knowledge to estimate the quality of workers for task assignment. While the quality based task assignment helps improve the aggregation accuracy, it provides no specific guarantee, e.g., the aggregated label of each task having at least 95% probability to match the ground truth. Subsequently, the unknown aggregation errors would propagate

to downstream applications of crowdsourcing. Another major issue needs to be addressed is worker cost, i.e., the cost for hiring online workers to label tasks. Though the payment for online workers in crowdsourcing is fairly cheap, as compared to the traditional way of outsourcing, it may still become unaffordable under a huge task number because multiple workers are normally hired for each task. In this case, heuristically choosing the budget without knowing the quality of online workers in advance may lead to an unnecessarily high cost, especially when fewer workers are already good enough for a desired accuracy on aggregated labels.

In this work, we consider three goals in the design of a crowdsourcing system. 1) The first goal is to *guarantee* a given aggregation accuracy δ for all completed tasks, i.e., the probability that the aggregated label agrees with the ground truth label is at least δ. 2) The second goal is to *minimize* the worker cost (i.e. number of workers assigned) under the δ guarantee. 3) Our third goal is to complete *as many tasks as possible* under the above two requirements, while the remaining tasks can be completed by any traditional algorithm without accuracy guarantee and cost minimization. Therefore, instead of completing all tasks with possibly unknown bad outcomes as in traditional methods, our approach ensures accuracy guarantee and cost minimization for as many tasks as possible.

If all workers and their quality are available in advance, we may estimate the probability for any set of workers to infer the true label based on worker quality and formulate the task assignment as a constrained optimization problem. Unfortunately, the online nature of crowdsourcing allows workers with unknown quality to become active (by joining the system) and inactive (by leaving the system) at any time, which makes it hard to determine the minimum worker cost for each task. In addition, many real crowdsourcing systems, such as Amazon Mechanical Turk (AMT) [1] and Figure Eight[2], and research works, such as *iCrowd* [26] and *Docs* [109], adopt the "request-then-assign" model where an active worker must first make a request, to indicate the readiness to work on the next task, before the system selects and assigns a task to the worker. In this case, when a worker will request is unknown in advance. Such practical natures of online settings pose the following challenges for solving our problem.

Challenges. While assigning a task to top workers, i.e., most qualified workers, can minimize the worker cost, we do not know when top workers will become active, so our cost minimization can only be defined w.r.t the top workers who are *currently* active. Even with active top workers, we do not know when they will request, therefore, we may need to assign a task to non-top workers who make a request earlier. This makes it hard to minimize the worker cost because in general more non-top workers are required than top workers to guarantee the aggregation accuracy. Therefore, how to meet a specified accuracy

[1] https://www.mturk.com/

[2] https://appen.com/figure-eight-is-now-appen/

guarantee δ with minimized worker cost w.r.t active top workers in the online setting is a challenging problem. This challenge is further compounded by the fact that active workers are dynamically changing.

Contributions. Our contributions are listed as follows:

1. We formalize a Guaranteed Accuracy Problem (GAP) (Section 3.3) to capture the requirement of completing as many tasks as possible with a guaranteed aggregation accuracy and minimized worker cost for the online setting described above.

2. As a key to complete more tasks for GAP, we formalize the notion of "candidate workers" (Section 3.5) who are as good as the top workers in providing a specified accuracy guarantee (Section 3.4) with minimized worker cost.

3. We present two solutions (Section 3.6), called Guaranteed Accuracy Algorithm (GAA) and iterative GAA ($IGAA$), to solve GAP by assigning tasks to candidate workers and completing additional tasks under the accuracy guarantee through the iterative relaxation of the minimum cost requirement.

4. We conducted empirical evaluation (Section 3.7) using real datasets to show that GAA and $IGAA$ can solve GAP efficiently.

3.2 Related Work

Our work aims to guarantee a given aggregation accuracy of completed tasks for the HIT requestor, and achieving this with the minimum worker cost. We review existing methods along this line by grouping them into three categories: accuracy improvement, cost reduction, and accuracy guarantee withminimized cost.

3.2.1 Accuracy Improvement

Most works in the literature focused on improving aggregation accuracy. For example, [19, 18, 98, 74, 60, 62, 96] first collect a fixed number of worker labels by random task assignment and then iteratively estimate worker quality with the inference of true labels using probabilistic graphical models. The difference lies on the modeling of worker accuracy [19, 98, 60], task difficulty [96, 62] and confusion matrix [18, 74] (i.e., the probability for a worker to provide a specific label on a task). The aggregation result can be further improved by assigning tasks according to each worker's accuracy on tasks, such as *iCrowd* [26] and *Docs* [109]. However, all these works do not address the issue of meeting an accuracy guarantee for label aggregation or minimizing the worker cost.

3.2.2 Cost Reduction

Research work also has been done on reducing worker cost [19, 92, 44, 93, 5, 86, 100] by removing unnecessary tasks whose aggregated labels can be inferred from other tasks. Such cost reduction considers the relationship between tasks, not the matching between workers and tasks like ours, so no aggregation accuracy can be guaranteed. Moreover, these methods consider specific types of tasks, e.g., entity resolution, to exploit domain-specific properties and structures.

3.2.3 Accuracy Guarantee with Cost Minimization

$CDAS$ [61] and $CQTO$ [1] reduce the worker cost by terminating the assignment of a task once a given confidence threshold on the aggregated label is satisfied. In both works, each worker is modeled by a single average accuracy for all tasks, which could be over-estimated for "harder" tasks, leading to the failure of meeting the threshold. In addition, the early termination of task assignment depends on the order of worker requests: if low-quality workers request early, more worker labels will be needed to strengthen the confidence score, so the cost will be high.

ATA [36] enforces a ϵ threshold on the aggregation error for binary classification tasks. Each task t is assigned to a minimum set of workers W_t with highest quality $q_{t,w}$ such that the quality sum $\sum_{w \in W_t} q_{t,w}$ passes a lower bound, i.e., $2ln(1/\epsilon)$, where $q_{t,w} = (2a_{t,w} - 1)^2$ and $a_{t,w}$ denotes worker w's accuracy for task t. Since this lower bound, based on Hoeffding's inequality, is loose, it can lead to an unnecessarily large worker cost. For example, while a single worker w with $a_{t,w} = 1$, i.e., 100% accuracy, is sufficient to achieve less than $\epsilon = 0.1$ error, the above inequality requires at least 5 such workers. In the online setting, even more worker cost would be needed because we cannot make sure each task is only assigned to workers with highest quality.

As we can see, existing works fail to provide a guarantee on aggregation accuracy due to modeling a single average accuracy for each worker, and fail to minimize the worker cost as it pertains to a specific worker request order (e.g., $CDAS$ [61] and $CQTO$ [1]) or a specific quality lower bound (e.g., ATA [36]). Our approach provides a guarantee for the aggregation accuracy by modeling the task-specific inference probability and our minimization of worker cost pertains to the set of active workers, not a particular worker request sequence.

3.3 Problem Definition

Task Group. A task group $T = \{t\}$ published by an HIT requester contains a number of human intelligent tasks (HITs), which are usually in an identical format but may require the knowledge in different domains. For example, a product labeling task group T may include 300 tasks about Apple iphones and 200 tasks about Samsung Galaxy tabs. Each task t is represented by a set of terms or a vector of features, and has L optional labels, one of which

is the true label but is unknown. To estimate worker quality, the HIT requester will provide a small set of *golden tasks* with known true labels, as denoted by $T' = \{t'\}$.

Online Workers. In the online setting, the active worker set $W = \{w\}$ is dynamic in that workers may join and leave the system at any time. To be assigned a task, a worker in W must first make a request, indicating that he is ready for the next task, then the system chooses an uncompleted task from T for the requesting worker. A task is *completed* once no more workers are needed for the task. For simplicity, we assume that the task assignment is non-preemptive, as applied in other crowdsourcing methods, e.g., *iCrowd* [26], meaning a worker must provide a label for an assigned task before requesting for the next one.

Table 3.1 summarizes the frequently used notations.

Table 3.1: Frequently used notations

Symbol	Description
$T = \{t\}$	normal tasks with unknown true labels
$W = \{w\}$	dynamic online workers
$T' = \{t'\}$	golden tasks with known true labels
L	the number of optional labels for a task
δ	the minimum aggregation accuracy expected by the HIT requester
λ	the maximum number of workers allowed on a task
W_t	the set of all workers assigned to a task t
$A(W_t)$	the probability for W_t to infer task t's true label
$\mathbb{A} = [a_{t,w}]$	a matrix of worker accuracy $a_{t,w}$ for each pair of worker w and task t
c_t^*	the minimum worker cost for a task t
a_t^*	the minimum worker accuracy for a task t

Definition 1 (Worker Accuracy)**.** *For each task t and each worker w, the worker accuracy $a_{t,w} \in [0, 1]$ denotes the probability for w to provide the true label of task t.*

Definition 2 (Inference Probability)**.** *Let W_t be the set of workers hired for a task t. The inference probability $A(W_t) \in [0, 1]$ denotes the probability for the label aggregated from those provided by workers in W_t to match the true label of t, where $A(W_t)$ could be estimated based on worker accuracy.*

With the online and dynamic nature of active workers, it may happen that some workers having higher accuracy for a task t join W after the task has been assigned to some other workers. Clearly, existing assignments cannot be undone in order to switch to the new workers. The best we can do is to ensure that each following assignment of the task is done to minimize the worker cost given the already assigned workers and the worker set W at the assignment time. In other words, the notion of cost minimization is time dependent

because already assigned workers and active workers are time dependent. This motivates the following problem definition.

Problem 1 (Guaranteed Accuracy Problem)**.** *Given parameters (δ, λ), we aim to assign requesting workers from the dynamic worker pool W to the tasks from T such that*

(i) For every task $t \in T$ that is completed, $A(W_t) \geq \delta$ and $|W_t| \leq \lambda$, where each task assignment is done to minimize the worker cost $|W_t|$ given the already assigned workers and the active worker set W at the assignment time.

(ii) The number of tasks completed under (i) is maximized for a given crowdsourcing time window. □

(i) ensures the quality guarantee for the aggregated label and the minimum worker cost for each completed task, where λ is the constant budget specified by the HIT requester to prevent from hiring too many workers for a task. Due to the (δ, λ) constraint in (i), it is possible that some tasks cannot be completed when the quality of workers in W is poor, so (ii) aims to maximize the number of completed tasks for a time window of processing specified by the HIT requester. Uncompleted tasks can always be completed by running an existing algorithm without quality guarantee or cost minimization.

The cost minimization for a task t, as indicated in (i), is dynamic in that it depends on the workers already assigned and the current set of active workers; however, it is independent of the requesting order of workers not assigned to the task yet. The latter is the major difference from the early termination in *CDAS* [61] and *CQTO* [1], where the cost minimization highly depends on the requesting order of online workers.

Example 2. *To explain the above points, consider an active worker set $W = \{w_1, w_2, w_3, w_4\}$ and a task t that can be completed jointly by $\{w_1, w_2, w_3\}$, but not any other subset of W, under the (δ, λ) requirement. Initially, no worker is assigned to t and the size of the subset $\{w_1, w_2, w_3\}$ is the minimum cost for t. Now, suppose that after w_1 has been assigned to tm denoted by $W_t^{assigned} = \{w_1\}$, a new worker w_5 with at least δ accuracy for t joins the system, i.e., the active worker set becomes $W = \{w_1, w_2, w_3, w_4, w_5\}$. Note that w_5 alone could complete t with the (δ, λ) requirement satisfied. However, given that the existing assignment $W_t^{assigned} = \{w_1\}$ cannot be undone, the minimum cost of t with respect to the new W will be 2 instead of 1, because $\{w_1, w_5\}$ can complete t under the (δ, λ) requirement.*

In general, the minimum cost for a task t is $|W_t^{assigned} \cup S|$, where S is a subset of $W \subset W_t^{assigned}$ with the minimum size such that $W_t^{assigned} \cup S$ can complete t while satisfying (δ, λ) the requirement. This notion of cost minimization for t is dynamic because it depends on $W_t^{assigned}$ and the active worker set W, both of which are dynamic. However, this notion does not depend on the requesting order of not assigned workers in $W \subset W_t^{assigned}$.

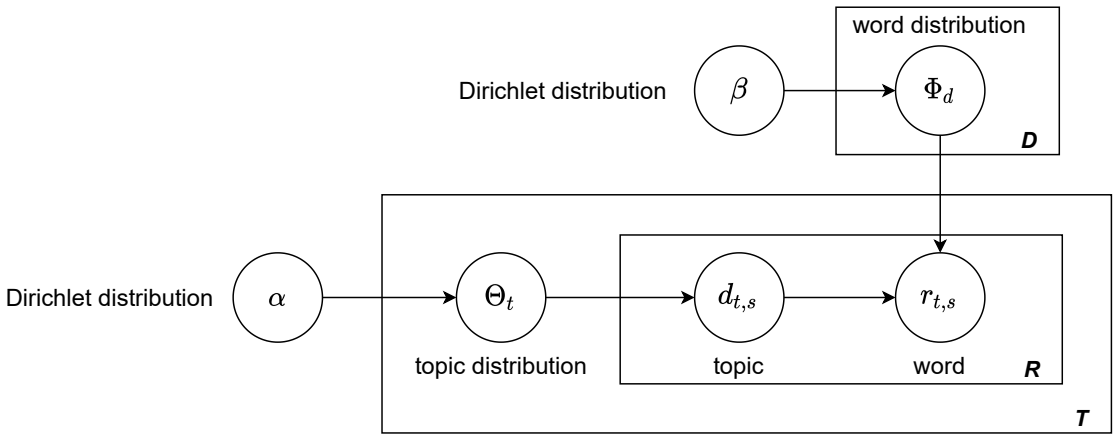

Figure 3.1: Latent Dirichlet Allocation (LDA)

3.4 Inference Probability Evaluation

In this section, we estimate the inference probability $A(W_t)$ in Problem 1, but before that, we first estimate the worker accuracy $a_{t,w}$, i.e., the probability for a worker w to provide the true label on a task t. In particular, with the assumption of D latent topics, we project tasks and workers into a topic domain, where each task t is represented by a topic distribution $\Theta_t = \{\theta_{t,1}, \cdots, \theta_{t,d}, \cdots \theta_{t,D}\}$ while each worker w is represented by a topic-wise accuracy $\Psi_w = \{\psi_{w,1}, \cdots, \psi_{w,d}, \cdots, \psi_{w,D}\}$.

Topic Distribution Θ_t. Recall that each task is represented by a set of terms or a vector of features. As suggested in *iCrowd* [26], a topic modeling technique called Latent Dirichlet Allocation (LDA) [9] can be applied to derive a D-dimension topic distribution $\Theta_t = \{\theta_{t,1}, \cdots, \theta_{t,d}, \cdots \theta_{t,D}\}$ for each task t, where $\sum_{d=1}^{D} \theta_{t,d} = 1$.

LDA is a topic modeling technique for uncovering hidden topics from text document. In our case, the text description of each task t could be treated as a document because the words used in the description usually reveal the topics of the task, provide a hint for workers to understand the topics, and help workers decide whether to work on the task or not. By encoding the probability of each particular word r w.r.t. each topic d into a R-dimension word distribution Φ_d, LDA algorithm first draws the prior Dirichlet distributions α and β for $Theta_t$ and Φ_d respectively, as shown in Figure 3.1. Let $d_{t,s}$ and $r_{t,s}$ denote the topic assignment and word assignment for the s^{th} word position in task t's text description. In each iteration, a topic $d_{t,s}$ is randomly chosen for a word position s based on $Theta_t$, followed by a word $r_{t,s}$ drawn from the corresponding Φ_d, where $d = d_{t,s}$. The posterior distribution could be inferred by Gibbs sampling [33]. After several iterations, the topic distribution Θ_t for each task t will be refined. Note that LDA can be performed *offline* before tasks are published. Topic modeling also makes it easy to handle a new task drawn from the same keyword distribution as the existing task collection, because the topic distribution can be derived on-the-fly from the current model.

Algorithm 1: *Acc_Estimate*$(w, T, T', \mathbb{A})$

Input: a new worker w, normal tasks T, golden tasks T', worker accuracy matrix $\mathbb{A}$
Output: updated worker accuracy matrix $\mathbb{A}$

```
test w using golden tasks in T';
compute Ψ_w using Equation 3.1;
for each task uncompleted task t ∈ T do
    compute a_{t,w} based on Θ_t and Ψ_w using Equation 3.2;
    update 𝔸 with worker w's accuracy a_{t,w} on task t;
return 𝔸;
```

Topic-Wise Accuracy Ψ_w. When a new worker w arrives, we ask the worker to label a set of golden tasks $T' = \{t'\}$ with known true label before assigning normal tasks to the worker. The purpose is to estimate the topic-wise accuracy Ψ_w of the worker. Typically, the use of golden tasks is to filter out workers that can not pass an overall accuracy rate, but we believe a worker should not be required to be good on all topics. Our method allows a worker to be incorrect in some topics but good at others. Then we can only assign the worker to those tasks that he may perform well.

Let $\mathbb{I}_{t',w}$ indicate whether a worker w's label agrees ($\mathbb{I}_{t',w} = 1$) or disagrees ($\mathbb{I}_{t',w} = 0$) with the true label of a task $t' \in T'$. For each topic d, $\psi_{w,d}$ can be estimated based on the weighed sum of $\theta_{t',d}$ for the golden tasks correctly labeled by worker w:

$$\psi_{w,d} = \textstyle\sum_{t' \in T'} (\theta_{t',d} \times \mathbb{I}_{t',w}) / \sum_{t' \in T'} \theta_{t',d} \tag{3.1}$$

where $\psi_{w,d} \in [0, 1]$. Since both the golden task size $|T'|$ and the topic size D are small constants, the running time for evaluating Ψ_w is $O(1)$.

Worker Accuracy $a_{t,w}$. With the topic distribution Θ_t of a task t and the topic-wise accuracy Ψ_w of a worker w, the accuracy $a_{t,w}$ of worker w on task t can be computed by the dot product of Θ_t and Ψ_w in a constant time:

$$a_{t,w} = \Theta_t \cdot \Psi_w = \textstyle\sum_{d=1}^{D} \theta_{t,d} \times \psi_{w,d} \tag{3.2}$$

$a_{t,w}$ ranges from 0 to 1, and $a_{t,w} = 1$ only if worker w has 100% accuracy for each related topic d, i.e., $\psi_{w,d} = 1$.

Algorithm 1 updates the worker accuracy matrix $\mathbb{A}$ for a new worker w by estimating the worker's accuracy $a_{t,w}$ on each uncompleted task $t \in T$. With Ψ_w computed in a constant time, the complexity of Algorithm 1 is $O(|T|)$.

Inference Probability $A(W_t)$. The estimated worker accuracy $a_{t,w}$ allows us to evaluate the inference probability $A(W_t)$ for a set of workers W_t on a task t. For example, under majority voting, $A(W_t)$ is the probability that the majority label of workers in W_t gives the true label of t (a tie is broken arbitrarily). Alternatively, we can consider the accuracy $a_{t,w}$

as the weight of worker t, and define the aggregated label as the label that receives highest total worker weight. For simplicity, we consider majority voting below.

Given any subset S of W_t, let $\Pr(S)$ be the probability that the workers in S provide the true label of t and the workers in $\bar{S} = W_t \setminus S$ provide a wrong label. From the Poisson binomial distribution,

$$\Pr(S) = \prod_{w \in S} a_{t,w} \prod_{w \in \bar{S}} (1 - a_{t,w}). \tag{3.3}$$

Let $m(S)$ be the probability that none of the $L-1$ wrong labels provided by workers in $W_t \setminus S$ receives at least $|S|$ votes, i.e., the true label provided by the workers in S is indeed a relative majority vote. Then $A(W_t)$ can be computed by

$$A(W_t) = \sum_{z=\lfloor \frac{|W_t|}{L} \rfloor + 1}^{|W_t|} \sum_{S \in \mathbb{S}_{t,z}} \Pr(S) m(S), \tag{3.4}$$

where $\mathbb{S}_{t,z}$ denotes all possible subsets S of W_t with $|S| = z$, and $\lfloor |W_t|/L \rfloor + 1$ is the smallest majority size. We note that $A(W_t)$ depends on $a_{t,w}$ for workers w in W_t, but not on the actual labels provided by the workers on t.

Let us compute $m(S)$. For any subset S of W_t with $|S| = z$, consider two cases:

Case 1: If $|W_t| - z < z$ (i.e., fewer workers provide the wrong labels than those providing the true label), we have $m(S) = 1$. Note that $L = 2$ falls into this case because $z \geq \lfloor \frac{|W_t|}{2} \rfloor + 1$ implies $|W_t| - z < z$.

Case 2: If $|W_t| - z \geq z$, $m(S) = 1 - a/b$, where a/b computes the probability that at least $|S| = z$ workers in $W_t \setminus S$ provide the same wrong label. Let's consider distributing the $|W_t| - z$ votes over the $L-1$ wrong labels. There are $b = (L-1)^{|W_t|-z}$ possible distributions in total. a is the number of distributions in which one of the $L-1$ wrong labels receives votes from at least z workers from $W_t \setminus S$ (i.e., those who provide a wrong label). $a = C_{L-1}^{1} \sum_{x=z}^{|W_t|-z} C_{|W_t|-z}^{x}$. So we have

$$m(S) = \begin{cases} 1 & \text{if } |W_t| - z < z \\ 1 - \frac{C_{L-1}^{1} \sum_{x=z}^{|W_t|-z} C_{|W_t|-z}^{x}}{(L-1)^{(|W_t|-z)}} & \text{otherwise} \end{cases} \tag{3.5}$$

Complexity Analysis. For $L = 2$, we have $m(S) = 1$, and $A(W_t)$ can be evaluated by a recursive computation [16] in $O(|W_t|^3)$ time, which is a constant as $|W_t|$ is bounded by the constant λ. For $L > 2$, $m(S)$ depends on $|W_t|$ and $|S| = z$, but not on S itself. With $|W_t| \leq \lambda$ and $\lfloor \frac{|W_t|}{L} \rfloor + 1 \leq z \leq |W_t|$, we can pre-compute $m(S)$ offline for z and $|W_t|$ in these ranges, and the complexity depends on the (usually small) constant λ. However, we still need to compute $\Pr(S)$ for each subset S with $|S| = z$ in the above range, and there are $|W_t|!/(|W_t|/2)!$ such subsets in total. In this case, the computation of $A(W_t)$ is less efficient, but since $|W_t| \leq \lambda$ and λ is usually a small constant, the computation time is a constant.

As the foundation for the concept of candidate workers defined in the next section, the following lemma indicates that replacing a worker in W_t with a better one would increase the inference probability $A(W_t)$.

Lemma 1 (Monotonicity of $A(W_t)$)**.** *Let W_t' be obtained from W_t by replacing a worker $w \in W_t$ with another worker w' such that $a_{t,w'} > a_{t,w}$. Then $A(W_t') > A(W_t)$.* □

Proof. Without loss of generality, we prove that increasing the accuracy a_{t,w_1} of a worker $w_1 \in W_t$ will increase $A(W_t)$. Recall that in Equation 3.3 and 3.4, $\Pr(S)$ denotes the probability that all workers in a subset S of W_t provide the true label and all workers in $W_t \setminus S$ provide a wrong label, and $m(S)$ denotes the probability that none of the wrong labels receive at least $|S|$ votes from the workers in $W_t \setminus S$ (i.e., the true label wins by receiving more votes than any wrong label). Depending on whether w_1 provides a true or wrong label, we have two cases:

Case 1: w_1 provides a true label, i.e., $w_1 \in S$. Increasing a_{t,w_1} will increase $\Pr(S)$ in Equation 3.3, which increases $\Pr(S)m(S)$ and thus $A(W_t)$ in Equation 3.4.

Case 2: w_1 provides a wrong label, i.e., $w_1 \in W_t \setminus S$. Increasing a_{t,w_1} will decrease $\Pr(S)$ in Equation 3.3, but we show that the subset $S' = S \cup \{w_1\}$ will compensate the decrease, that is, the total contribution of $\Pr(S)m(S)$ and $\Pr(S')m(S')$ to $A(W_t)$ in Equation 3.4 will not decrease by increasing a_{t,w_1}.

$$
\begin{aligned}
&\Pr(S)m(S) + \Pr(S')m(S') \\
&= m(S) \prod_{w \in S} a_{t,w} \prod_{w \in W_t \setminus S} (1 - a_{t,w}) \;+\; m(S') \prod_{w \in S'} a_{t,w} \prod_{w \in W_t \setminus S'} (1 - a_{t,w}) \\
&= m(S)(1 - a_{t,w_1}) \prod_{w \in S} a_{t,w} \prod_{w \in W_t \setminus S'} (1 - a_{t,w}) \;+\; m(S') a_{t,w_1} \prod_{w \in S} a_{t,w} \prod_{w \in W_t \setminus S'} (1 - a_{t,w}) \\
&= (m(S) + (m(S') - m(S)) a_{t,w_1}) \prod_{w \in S} a_{t,w} \prod_{w \in W_t \setminus S'} (1 - a_{t,w}).
\end{aligned}
$$

Increasing a_{t,w_1} only affects $(m(S') - m(S))a_{t,w_1}$. We show $m(S') - m(S) > 0$, thus, increasing a_{t,w_1} increases $\Pr(S)m(S) + \Pr(S')m(S')$, and thus $A(W_t)$.

Let n_S denote all the distributions of votes for workers in $W_t \setminus S$ such that none of the $L-1$ wrong labels receive at least $|S| = z$ labels. From Equation 3.5, $m(S) = |n_S|/(L-1)^{(|W_t|-z)}$ and $m(S') = |n_{S'}|/(L-1)^{(|W_t|-z-1)} = (L-1)|n_{S'}|/(L-1)^{(|W_t|-z)}$. For each distribution in n_S, we can form a corresponding distribution in $n_{S'}$ by removing w_1's vote from the distribution, which means $|n_{S'}| > |n_S|$ and $m(S') - m(S) > 0$, as claimed. □

3.5 Candidate Worker Identification

According to Lemma 1, assigning a task to its top workers with highest accuracy can minimize the worker cost. However, in the online setting we do not know when top workers

will make a request. To address this issue, our approach is identifying "candidate workers", for each task t, that are *as good as* top workers in the sense that if k top workers can satisfy the (δ, λ) requirement with the minimum worker cost, so do any k candidate workers. For example, suppose $\{w_1, w_2, w_3\}$ is the minimum set of top workers on a task t for achieving at least δ inference probability, so $k = 3$. We could define all workers in $\{w_1, w_2, w_3, w_4, w_5, w_6\}$ as candidate workers if any 3 of them can pass the δ threshold for t and complete the task whenever any 3 of them make a request, without waiting for the top workers $\{w_1, w_2, w_3\}$.

In this section, we define and identify candidate workers for a task t by two parameters (c_t^*, a_t^*). c_t^* is the *minimum worker cost* for meeting the accuracy threshold δ on t. a_t^* is the *minimum worker accuracy* such that for any set W_t of c_t^* workers w with $a_{t,w} \geq a_t^*$, we have $A(W_t) \geq \delta$. We first consider a simpler case of static W in Section 3.5.1 and then extend to dynamic W in Section 3.5.2.

3.5.1 Candidate Workers under Static Worker Set W

To compute c_t^* with a static W, we need to find the minimum set W_t^m of workers w in W with highest accuracy $a_{t,w}$ such that $A(W_t^m) \geq \delta$ and $|W_t^m| \leq \lambda$. Based on Lemma 1, $|W_t^m|$ is the smallest number of workers required to meet these conditions, so $c_t^* = |W_t^m|$. Since $A(\cdot)$ is computed in a constant time (Section 3.4), the complexity of computing c_t^* is $O(|W|)$. If this W_t^m does not exist, let $c_t^* = \infty$.

Example 3. *Assume $\delta = 0.9$, $\lambda = 5$, and the current active workers in $W = \{w_1, ..., w_7\}$ have 0.88, 0.86, 0.84, 0.83, 0.82, 0.81 and 0.75 accuracy on a task t, respectively. For simplicity, consider $L = 2$. $W_t^m = \{w_1, w_2, w_3\}$ is the minimum set of top workers for meeting the δ requirement because Equation 3.4 gives $A(\{w_1\}) = 0.88 < \delta$, $A(\{w_1, w_2\}) = 0.76 < \delta$ and $A(\{w_1, w_2, w_3\}) = 0.946 \geq \delta$. Thus, we have $c_t^* = 3$.*

To compute a_t^*, we can construct an "virtual" worker set W_t^a (not actual workers in W) containing c_t^* workers having the same accuracy a on task t. Based on Lemma 1, a_t^* is equal to the minimum value a such that $A(W_t^a) \geq \delta$. To find this minimum value a, we can apply a binary search over the interval $(0, 1]$ and, according to Lemma 1, prune the half that fails the condition $A(W_t^a) \geq \delta$ in each iteration. For example, with the precision of 0.01, there are 100 discrete points in the interval. With $O((c_t^*)^3)$ time required for evaluating $A(W_t^a)$ (Section 3.4), the complexity of the binary search is $O((c_t^*)^3 \log_2 100)$, which is constant as $c_t^* \leq \lambda$. If $c_t^* = \infty$, let $a_t^* = \infty$.

Example 4. *With the minimum cost $c_t^* = 3$ in Example 3, we can create a size-3 virtual worker set W_t^a, where all workers in W_t^a have the same accuracy a on task t. Then, a binary search of a is performed over $(0, 1]$. In each iteration, a is set to the mid value of the remaining interval, e.g., 0.5 in iteration 1, and the larger half interval is pruned if $A(W_t^a) \geq \delta$ or the smaller half interval is pruned otherwise. When the precision reaches 0.01, we have $a_t^* = a = 0.81$.* □

Algorithm 2: $Cand\_Worker(t, W, \delta, \lambda, \mathbb{A})$

Input: a task t, current active worker set W, minimum aggregation accuracy δ, maximum worker cost λ, worker accuracy matrix $\mathbb{A}$

Output: (c_t^*, a_t^*): minimum worker cost and minimum worker accuracy of task t

$c_t^* \leftarrow \infty, a_t^* \leftarrow \infty$, $W^{assigned} \leftarrow$ workers assigned for task t;

Find shortest prefix W_t^m of top workers $w \in W \setminus W^{assigned}$ with highest accuracy $a_{t,w} \in \mathbb{A}$ such that $A(W^{assigned} \cup W_t^m) \geq \delta$ and $|W^{assigned} \cup W_t^m| \leq \lambda$;

if W_t^m *can be found* **then**

  $c_t^* \leftarrow |W^{assigned} \cup W_t^m|$;

  $W_t^a \leftarrow c_t^* - |W^{assigned}|$ virtual workers with accuracy a;

  Binary search the minimum a over $(0.5, 1]$ such that $A(W^{assigned} \cup W_t^a) \geq \delta$ until the precision reaches 0.01;

  $a_t^* \leftarrow a$;

return (c_t^*, a_t^*)

Definition 3 (Candidate Workers)**.** *For an uncompleted task t, a worker w from W is a candidate worker if t has not been assigned to w and $a_{t,w} \geq a_t^*$.* □

Following the definitions of c_t^* and a_t^*, the next theorem states that candidate workers are as good as top workers in guaranteeing the δ accuracy with the minimum worker cost.

Theorem 1. *Assume $c_t^* \leq \lambda$. Let W_t be any set of c^* candidate workers of t (i.e., $a_{t,w} \geq a_t^*$). Then $A(W_t) \geq \delta$.*

Example 5. *Continue with Example 3 and 4. All workers in W except for w_7 are candidate workers for t as they have at least $a_t^* = 0.81$ accuracy. Therefore, any $c_t^* = 3$ of $\{w_1, ..., w_6\}$ can achieve at least δ inference probability and 3 is minimum. In particular, even if none of the top three workers $\{w_1, w_2, w_3\}$ requests, we can assign t to any 3 candidate workers who request, say $\{w_4, w_5, w_6\}$, and this still guarantees the $\delta = 0.9$ aggregation quality and the minimum cost.* □

3.5.2 Handling Dynamic Worker Set W

Now we consider the case that W changes to W' after some workers $W_t^{assigned}$ have been assigned to an uncompleted task t. In this case, (c_t^*, a_t^*) computed based on W will be out-dated and should be recomputed based on W', under the constraint that the workers in $W_t^{assigned}$ are already assigned to t. In particular, in the computation of c_t^*, W_t^m would become $W_t^{assigned} \cup W_t^m$ with W_t^m being selected from $W' \setminus W_t^{assigned}$, and $c_t^* = |W_t^{assigned} \cup W_t^m|$. Similarly, in the computation of a_t^*, W_t^a would become $W_t^{assigned} \cup W_t^a$ with W_t^a containing $c_t^* - |W_t^{assigned}|$ virtual workers with the same accuracy a, and a_t^* is the minimum value a found by the binary search such that $A(W_t^{assigned} \cup W_t^a) \geq \delta$.

Algorithm 2 shows the function $Cand_Worker$ for updating the parameters (c_t^*, a_t^*) that define candidate workers of a task t, which takes $O(|W|)$ time as discussed in Section 3.5.1.

Algorithm 3: $GAA(T, T', \delta, \lambda)$

Input: normal tasks T, golden tasks T', minimum inference probability δ, maximum worker cost λ

Output: aggregated labels of completed tasks

$T^c \leftarrow \emptyset$, $W \leftarrow \emptyset$, $W^{new} \leftarrow \emptyset$, $W^{leave} \leftarrow \emptyset$, $\mathbb{A} \leftarrow \emptyset$
while $|T^c| \neq |T|$ and within specified time window **do**
  switch (worker activity)
  case 1. Worker w requests**:**
    Assign a task $t \in T \setminus T^c$ with highest $a_{t,w} \in \mathbb{A}$ such that $w \notin W_t$ and $a_{t,w} \geq a_t^*$
    Add t to the completed task set T^c if $|W_t| = c_t^*$
  case 2. a new worker w arrives**:**
    $\mathbb{A} \leftarrow Acc\_Estimate(w, T, T', \mathbb{A})$
    Add w to W^{new}
  case 3. a worker w becomes inactive**:**
    Add w to W^{leave}
  case 4. W^{new} or W^{leave} is non-empty for a specified time interval**:**
    $W \leftarrow (W \cup W^{new}) \setminus W^{leave}$
    $W^{new} \leftarrow \emptyset$, $W^{leave} \leftarrow \emptyset$
    $(c_t^*, a_t^*) \leftarrow Cand\_Workers(t, W, \delta, \lambda, \mathbb{A})$ for each task $t \in T \setminus T^c$
Aggregate collected labels for completed tasks T^c by majority voting;
return aggregated labels of completed tasks T^c;

With the update of (c_t^*, a_t^*) on a dynamic worker set W, Definition 3 remains the same if W refers to the current active worker set, and Theorem 1 now is generalized as follows.

Theorem 2 (Dynamic W)**.** *Assume $c_t^* \leq \lambda$. Let $W_t^{assigned}$ be the workers already assigned to t, let W_t^c be any set of $c_t^* - |W_t^{assigned}|$ candidate workers of t, and let $W_t = W_t^{assigned} \cup W_t^c$. Then $A(W_t) \geq \delta$.*

According to Theorem 2, given the assigned workers, considering candidate workers for the remaining assignment will minimize the worker cost (because the cost c_t^* is minimized for the remaining assignment).

3.6 Online Assignment Algorithms

We now present the Guaranteed Accuracy Algorithm (GAA) and the iterative GAA ($IGAA$) for assigning candidate workers to tasks in the online setting.

3.6.1 Guaranteed Accuracy Algorithm (GAA)

$GAA(T, T', \delta, \lambda)$ in Algorithm 3 takes normal tasks T, golden tasks T', and (δ, λ) as input, and outputs the aggregated labels for completed tasks in $\mathcal{L}$. It repeatedly responds to the worker activity in four cases until either all tasks are completed (i.e., $\mathcal{L} = |T|$) or the specified time window for processing has passed.

Case 1 assigns a task t to a requesting candidate worker w, checked by the conditions $w \notin W_t$ and $a_{t,w} \geq a_t^*$. Theorem 2 implies that Case 1 ensures tasks are completed by satisfying the accuracy threshold δ with minimized worker cost c_t^*. Case 2 tests new workers using golden tasks T' and update $a_{t,w}$ for uncompleted tasks $t \in T \setminus T^c$, while Case 3 handles workers leaving the system. To prevent updating candidate workers too frequently, new workers and leaving workers are held up in W^{new} and W^{leave} until a specified time interval has passed before W and (c_t^*, a_t^*) for candidate workers are updated in Case 4. The time interval serves as a trade-off between the update timeliness of W and the frequency of calling $Cand\_Worker(t, W, \delta, \lambda, \mathbb{A})$.

Note that the task assignment is assumed to be non-preemptive for simplicity in Section 3.3, where a worker must provide a label for an assigned task before requesting for the next one. But in reality, a worker may leave the system at any time, even without labeling the assigned task. To cope with this situation, we could suspend the assignment of a task t when the number of workers in W_t who are assigned to the task reaches c_t^* and add the task to the completed task set T^c in Algorithm 3 only when all these c_t^* workers have provided their labels. If a worker in W_t fails to provide a label within a predefined time limit, we can remove the worker from W_t and resume the assignment of the task.

Complexity Analysis. With recorded c_t^*, a_t^* and $a_{t,w}$, Case 1 takes $O(|T|)$ time. Case 2 requires $O(|T|)$ time to estimate $a_{t,w}$ for all uncompleted tasks. The complexity of Case 3 is $O(1)$. For Case 4, $Cand\_Worker(t, W, \delta, \lambda, \mathbb{A})$ can be done in $O(|W|)$ time for each uncompleted task t (see Section 3.5.1), so the update cost for Case 4 is $O(|T||W|)$. This cost is smaller than the update cost $O(|T|^2 + |T||W|)$ of *iCrowd* [26] that is incurred for each change of W and each submission of worker labels.

We note that a high-quality worker is likely candidate workers for many tasks in GAA, but it would not overload such workers in terms of assigning the worker too many tasks. This is because under the "request-then-assign" model, a task is assigned to a worker only if the worker makes a request, which indicates that the worker is ready to take the next task. Actually, making more use of high-quality workers is the key to improve the aggregation accuracy.

3.6.2 Iterative GAA ($IGAA$)

With Algorithm 3, it is possible that some tasks may be left uncompleted after the specified processing time window expires. This happens if the δ threshold is set unrealistically high relative to the quality of the workers in W, or the top workers for a task t have such a high accuracy that it causes a very small c_t^* and a very high a_t^*, so not many requesting workers qualify to be a candidate worker for the task.

We can address the above issue by running GAA in multiple iterations. In iteration i, initially $i = 0$, the minimum worker cost $c_t^*(i)$ is set to $min(c_t^* + i, \lambda)$ for uncompleted tasks t in iteration i, where $c_t^* \neq \infty$. In other words, the minimum worker cost is gradually relaxed

by 1 in each next iteration. This iterative process terminates in three cases: all tasks are completed, or the HIT requester decides to terminate, or $c_t^*(i) \geq \lambda$ for every uncompleted task t at the end of iteration i. The last case indicates that the minimum worker cost cannot be further relaxed because $|W_t| \leq \lambda$ is required. We call this algorithm *Iterative GAA* or simply *IGAA*. Note that *GAA* is *IGAA* with only the initial iteration. The complexity of each iteration for *IGAA* is similar to that of *GAA*, except that the number of uncompleted tasks shrinks after each iteration.

Like *GAA*, *IGAA* ensures at least δ accuracy for all completed tasks. The difference is that the minimum worker cost requirement is relaxed by one in each next iteration to complete more tasks. If there are still uncompleted tasks left when the algorithm terminates, a traditional algorithm, such as hiring a fixed number of workers per task, can be applied to complete the remaining tasks without quality guarantee.

3.7 Experiment

We explain experiment settings (Section 3.7.1), study the benefits of candidate workers (Section 3.7.2), compare *GAA* and *IGAA* with state-of-the-art methods (Section 3.7.3 and 3.7.4). All experiments were implemented in Java on a computer with CPU 3.60GHz and 32GB memory.

3.7.1 Experiment Settings

Evaluation Criteria. We consider three evaluation criteria for solving *GAP*. ***Accuracy***: the percentage of completed tasks whose aggregated labels match true labels, which is the empirical inference probability. An algorithm produces a valid solution if Accuracy is at least δ. ***Cost***: the average worker number per completed task. ***Completion***: the percentage of tasks that are completed by an algorithm.

Experiments on a real platform like AMT will collect the data only with a particular worker requesting order, which may not lead to a conclusive performance study as we want to evaluate how different methods perform with respect to different worker requesting orders. For this reason, we decided to employ the following two crowdsourcing datasets previously collected so that we can test 50 randomly permuted orders of worker requests.

MovieLens[3] (ML). This dataset contains user ratings on movies and was used in previous crowdsourcing methods [83, 107]. Following the practice in [17], we view rating movies as binary labeled tasks, Yes for rating ≥ 3 and No for rating < 3. We removed movies with less than 7 ratings due to insufficient labels for testing. The resulting data has 3025 movies (tasks) and 668 users (workers). The rating recommended by the website was treated as the true label. The dataset also provides genre tags (19 genres in total) for each

[3]http://grouplens.org/datasets/movielens/

movie. A movie could have several genre tags. Since genres reflect the topics of a move, we directly derive the topic distribution of tasks from the genre information by equally weighting each related genre tag, instead of applying the LDA algorithm. For example, a movie with "Action", "Animation", "Children" and "Mystery" tags will have 0.25 weight for each of these 4 tags and 0 weight for all other tags.

Narrativity[4] **(NR).** This dataset collected by [35] asks 155 workers to review the abstracts of 802 scientific papers (tasks) and provide a Yes or No label to indicate whether the narrator refers to the author himself in the abstract (e.g., through use of pronouns such as I, we, and our). For each task, exactly 7 worker labels were collected from AMT. The true label for each task was manually obtained. As the abstract usually reveals the topics of a paper, the latent topic distribution of tasks is calculated using LDA. We set the number of latent topics to 10 and take the keywords in the abstract as the input for the topic modeling algorithm.

Both ML and NR datasets provide the $< t, w, l >$ tuples for the label l given by a worker w on a task t. For each dataset, we report the average of 50 runs. For each run, we generated the worker requesting sequence $w_1, \cdots, w_n$ from a random order of the tuples, $< t_1, w_1, l_1 >, \cdots, < t_n, w_n, l_n >$. Each algorithm assigns a task t to a requesting worker w according to its assignment strategy, with the constraint that the tuple $< t, w, l >$ for some label l is contained in the dataset. For each run, the golden tasks T' are created by sampling 20 $<t, w, l>$ tuples (with replacement) for each worker w for testing the worker. A worker on his first request is treated as a new worker and a worker after submitting his last label with no more requests is treated as leaving the system.

Baseline Methods. We compare the performance of our *GAA* and *IGAA* algorithms for solving *GAP* under the (δ, λ) constraint with five state-of-the-arts baseline methods:

(1) *iCrowd* [26]: This work greedily schedules the assignment of each task to a fixed number of top workers and each worker can be scheduled for at most one task at a time. The schedule is updated whenever W changes or a worker becomes available after submitting a label. After collecting worker labels, weighted majority voting is used to derive the aggregated labels. We set the fixed worker cost to λ (similarly, for *Docs* and *FaitCrowd* below).

(2) *Docs* [109]: This algorithm fixes the total worker cost for all tasks and assigns each task to a different number of workers according to the ambiguity of aggregated labels and the quality of requesting workers, which are iteratively updated using the EM algorithm. We set the total worker cost to $\lambda \times |T|$, where $|T|$ is the number of tasks (so that each task is labeled by λ workers on average).

[4]https://data.world/crowdflower/narrativity-in-scientific-pub

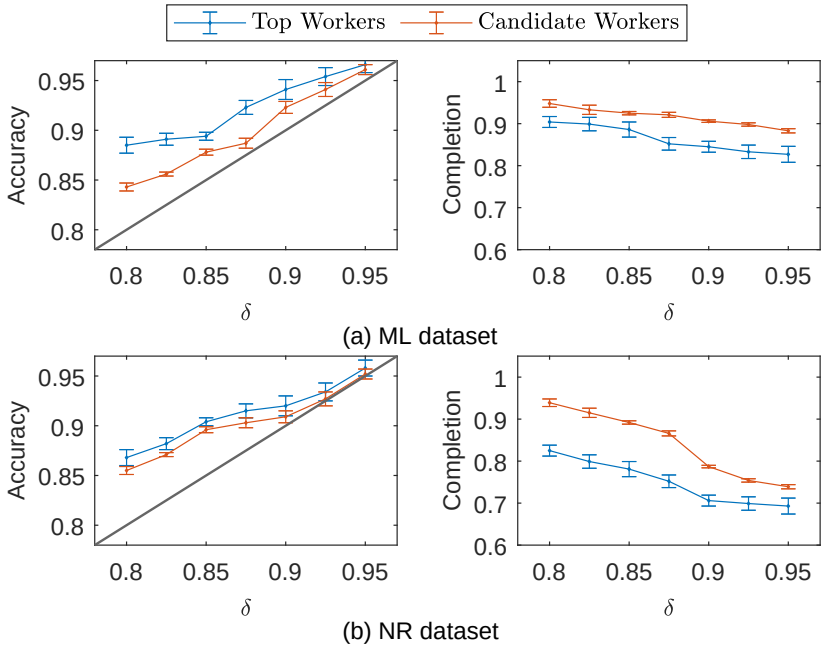

Figure 3.2: Accuracy and Completion (mean and standard error) by top workers vs candidate workers.

(3) *FaitCrowd* [62]: $FaitCrowd$ randomly assigns each task to λ workers and then applies a probabilistic graphical model to iteratively estimate worker expertise and task difficulty on the topic domain with the inference of true labels.

(4) *CDAS* [61]: $CDAS$ associates each worker with a single accuracy for all tasks, so an arbitrary task is assigned to requesting workers. The aggregated label is determined by a confidence value evaluated based on worker accuracy, and the assignment of a task is terminated when the confidence on the current best aggregated label passes a threshold derived from a given aggregation accuracy. We set this aggregation accuracy to δ.

(5) *ATA* [36]: This is the offline solution of ATA, which optimally assigns a minimum set of workers with highest quality to each task until the $2\ln(1/\epsilon)$ threshold on the quality sum is met, where ϵ is a specified aggregation error and we set $\epsilon = 1 - \delta$. The aggregated labels are determined by weighted majority voting. Note that this offline solution provides an upper bound on aggregation accuracy and a lower bound on worker cost for their online ATA.

Note that *iCrowd*, *Docs* and $FaitCrowd$ do not enforce δ and assign each task to λ workers on averages.

3.7.2 Benefits of Candidate Workers

We first study the benefits of candidate workers relative to top workers in terms of boosting of task completion rate. In Figure 3.2, we compare Accuracy and Completion of GAA

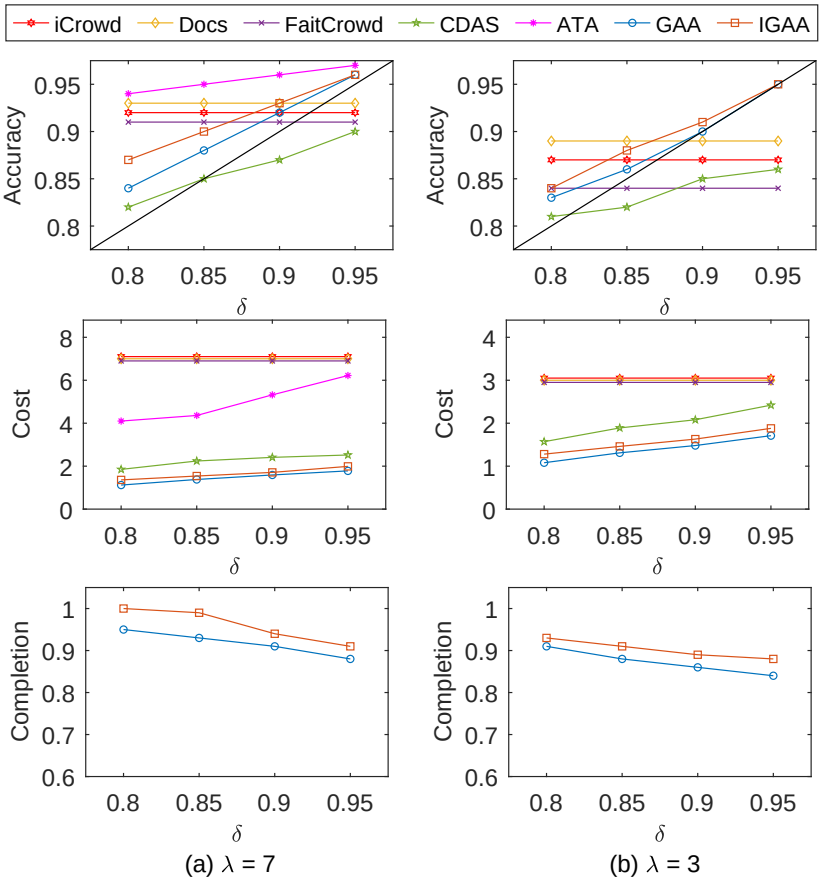

Figure 3.3: Accuracy, Cost and Completion for $\delta \in \{0.8, 0.85, 0.9, 0.95\}$ and $\lambda \in \{3, 7\}$ on ML dataset.

algorithm with two options. $CandidateWorkers(CW)$ denotes the option of assigning a task to its candidate workers exactly in Algorithm 3, and $TopWorkers(TW)$ denotes the option of assigning a task to its top workers instead. Cost is the same in the two cases. While TW has higher Accuracy than CW, both algorithms satisfied Accuracy $\geq \delta$, by staying above the diagonal line. However, CW helps complete 5% to 15% more tasks, which confirms the benefits of replacing top workers with candidate workers.

3.7.3 Comparison with Baselines

We first compare GAA and $IGAA$ with the baselines. In Case 4 of Algorithm 3, W and candidate workers are updated whenever a new worker arrives or an existing worker leaves. The processing time window is set to cover all worker requests. In Figure 3.3 and 3.4, we report the Accuracy, Cost and Completion of each algorithm on ML and NR, respectively, where $\lambda \in \{3, 7\}$ ((a) and (b)) and $\delta \in \{0.8, 0.85, 0.9, 0.95\}$ with the default settings of $\delta = 0.9$ and $\lambda = 7$.

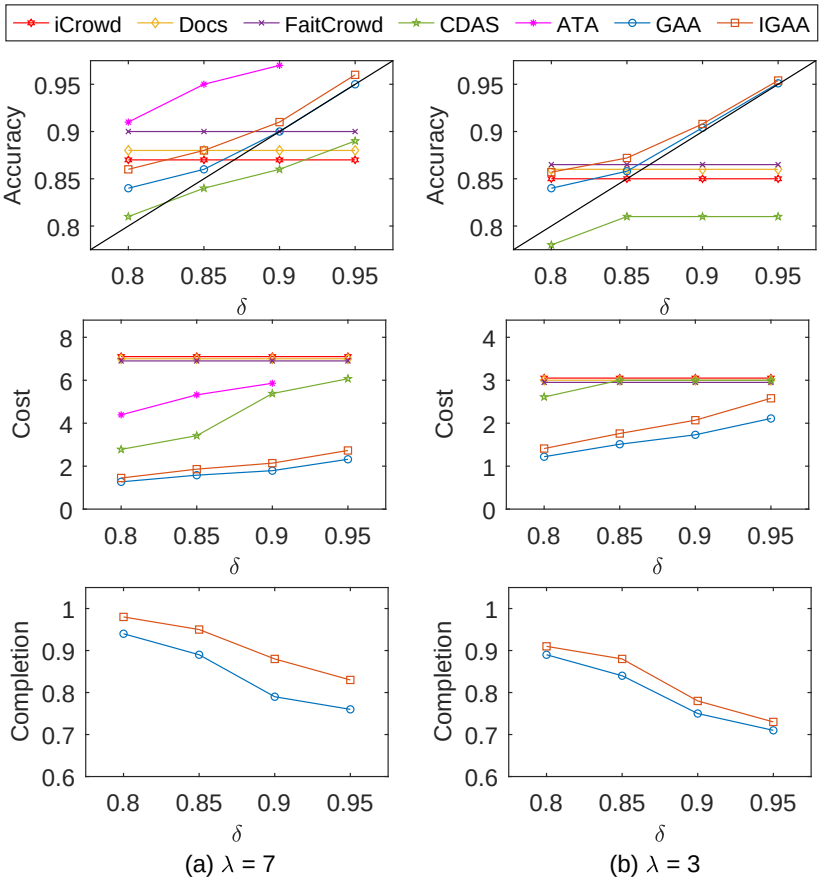

Figure 3.4: Accuracy, Cost and Completion for $\delta \in \{0.8, 0.85, 0.9, 0.95\}$ and $\lambda \in \{3, 7\}$ on NR dataset.

Accuracy. A violation of the δ accuracy requirement is indicated by falling below the diagonal line. For a large δ threshold, *iCrowd*, *Docs* and *FaitCrowd* could violate the δ requirement, especially when λ is small, e.g., $\lambda = 3$. *CDAS* responds to the change of δ, but since a *single* average accuracy is used to model each worker, which may be over-estimated for difficult tasks, the method fails to pass most δ thresholds. For $\lambda = 7$, we cannot find a solution for *ATA* on the NR dataset when $\delta = 0.95$ because no task can satisfy its lower bound on the quality sum. If we reduce λ to 3, *ATA* will fail for all δ settings tested. In contrast, our *GAA* and *IGAA* methods satisfy all (δ, λ) requirements on both datasets. All algorithms have a standard error ranging from 0.01 to 0.03.

Cost. Our *GAA* and *IGAA* algorithms hire much fewer workers per completed task than the baselines. In most cases, less than 2 workers per completed task are needed, and many tasks were completed with only 1 worker. Cost is reduced by 28% to 57% as compared to *CDAS*, by 63% to 78% as compared to *ATA*, and by 55% to 80% as compared to *iCrowd*, *Docs* and *FaitCrowd*. *ATA* requires even more Cost than *CDAS*. *ATA* compared here is the offline version. The online *ATA* will have lower Accuracy and higher Cost. Also, the

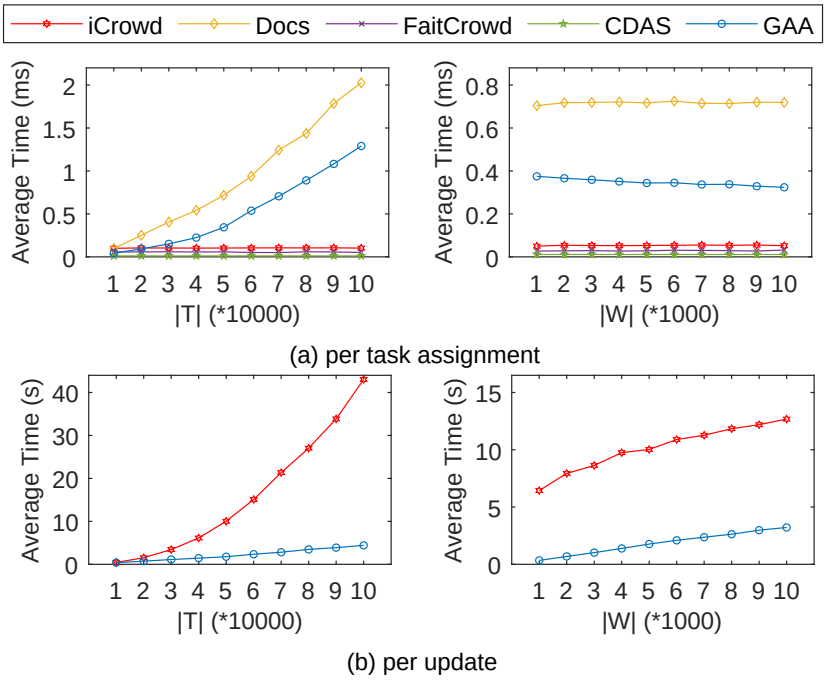

Figure 3.5: Average time per task assignment (a) and average time per update (b).

standard error of $CDAS$ (0.27) is much higher than other methods (0.00 to 0.05) because the cost minimization of $CDAS$ highly depends on the requesting order of workers.

Completion. $iCrowd$, $Docs$, $FaitCrowd$, $CDAS$ and ATA completes all tasks but as discussed above this is at the expense of no quality guarantee. GAA completes at least 88% tasks on the ML dataset and at least 76% tasks on the NR dataset when $\lambda = 7$, and all completed tasks satisfy the δ accuracy requirement. $IGAA$ increases Completion by 3% to 5% on the ML dataset and 5% to 9% on the NR dataset. When the smaller $\lambda = 3$ is applied, Completion of $GAA/IGAA$ on both datasets drops by only 4% to 7% as compared to $GAA/IGAA$ with $\lambda = 7$, because reducing λ from 7 to 3 only affects the completion of tasks that require more than 3 workers, which are very few. Note that the remaining tasks can be completed by applying a traditional method with no guarantee on the quality of aggregation results. The standard error of $GAA/IGAA$ ranges from 0.02 to 0.04.

3.7.4 Execution Time

We compare the efficiency of GAA with the online task assignment baselines, i.e., $iCrowd$, $Docs$, $FaitCrowd$ and $CDAS$, w.r.t. the task size $|T|$ from 10,000 to 100,000 and the worker size $|W|$ from 1,000 to 10,000. The default sizes of $|T|$ and $|W|$ are 50,000 and 5,000, respectively. By varying one of $|T|$ and $|W|$ while holding the other at the default size, we generated several synthetic datasets as follows: Each task is associated with a randomized topic distribution of length 5 and a randomized binary true label. For each worker w, a

random accuracy $a_w \in [0, 1]$ is associated so that the worker has a_w probability to provide the true label. The requesting order of workers is also randomized. As before, the golden task size $|T'|$ is set to 20.

First, we focus on the average task assignment time to respond to each worker request. For GAA, this is the time for each execution of Case 4 of Algorithm 3. Figure 3.5(a) shows the average task assignment time. $iCrowd$ schedules the task assignment before worker requests, while $FaitCrowd$ and $CDAS$ randomly choose a task to assign, so they all take little time for task assignment. $Docs$'s and GAA's task assignment time linearly increases with the task size $|T|$. Overall, all methods respond to each worker request in a few milliseconds, even for the task size of 100,000. The time for task assignment is independent of the worker size.

Next, we consider the average time per update required to maintain the information used by task assignment. Note that $FaitCrowd$, $CDAS$, and $Docs$ have little or no update time because they assign a randomly chosen task or the most benefit task. For GAA, this is the time for each execution of Case 4 of Algorithm 3 update the parameters of candidate workers when W changes. For $iCrowd$, this is the time on updating the task assignment schedule for each worker when a worker joins/leaves the system or submits a label.

As shown in Figure 3.5(b), $iCrowd$ has a much larger average update time than GAA. This is because the time of updating the task assignment schedule is $O(|T|^2 + |T||W|)$ [26], compared to $O(|T||W|)$ for updating candidate workers in GAA. In addition, $iCrowd$ has more frequent update than GAA because its update is triggered by each label submission of workers and the change of W, whereas GAA's update is triggered only by the change of W.

3.8 Summary

In this chapter, we analyze the drawbacks of existing works related to quality control and cost management in crowdsourcing. Previous works either do not address a specified aggregation accuracy, which is not acceptable to high stake downstream applications, or fail to minimize the cost of hiring online workers. We address this issue by presenting a new problem statement, called Guaranteed Accuracy Problem (GAP), with the dual requirements of accuracy guarantee and cost minimization.

To solve the problem under the online setting, where the set of active online workers is dynamically changing, we present a GAA framework to adaptively identify and update candidate workers who are as good as the top workers in providing a specified accuracy guarantee with minimized worker cost. An iterative version of GAA, named $IGAA$, is also proposed to complete more tasks by relaxing the cost minimization in each iteration.

Empirical evaluation on real crowdsourcing datasets confirm that $GAA/IGAA$ algorithms meet the design goals on accuracy guarantee and minimized cost. More specific,

GAA/*IGAA* algorithms pass all the different thresholds for aggregation accuracy and require much less cost than the state-of-the-art methods. We also create synthetic datasets with varying task number and worker number to show that our methods can effectively perform task assignment and update candidate workers w.r.t. the change of active online workers.

Chapter 4

TDSSA: Truth Discovery against Strategic Sybil Attack

Typical crowdsourcing methods aggregate worker labels in a voting manner, with the assumption that workers provide their labels independently. However, such aggregation is vulnerable to Sybil attack where the attacker earns easy rewards by coordinating several Sybil worker accounts to share a randomized label on each task for dominating the aggregation result. A strategic Sybil attacker may also attempt to evade Sybil detection. In this chapter, we propose a model called *TDSSA* to defend against strategic Sybil attack by integrating the Sybil behavior and label reliability of workers into truth discovery and well camouflaging tasks that are used to discover these worker properties.

4.1 Background and Overview

Crowdsourcing is often used as a tool for social efforts, e.g., the collection and analysis of data relating to the natural world by members of the general public, typically as part of a collaborative project with professionals. As workers with diverse knowledge background may provide conflicting labels to the same task, some number K of workers are hired for each task and their labels are aggregated to infer the unknown true label in a voting manner, such as by majority voting [26], weighted voting [36] and Bayesian voting [111]. An underlying assumption is that all workers provide labels independently and honestly, which fails in the presence of *Sybil attack* [23].

In crowdsourcing, a Sybil attacker could manipulate several Sybil worker accounts to share a randomized label on each task so that the independent workers on the same task could be outvoted. Sybil attack becomes *strategic* and is harder to defend if the attacker attempts to evade detection [68]. For example, if golden tasks are leveraged to each new worker in a cold-start phase for estimating worker quality, a strategic Sybil attacker can first label golden tasks honestly to improve the trustworthiness of Sybil workers but attack later on normal tasks. Even if we mix the assignment of golden tasks and normal tasks, the

Task	True Label	Worker Label					Aggregated Label
		w_1	w_2	w_3	w_4	w_5	
t_1	1	1	1	2	2	2	2
t_2	3	3	1	3	3	1	3
t_3	3	3	3	2	2	2	2
t_4	2	2	2	2	3	3	2
t_5	2	2	2	1	1	1	1

Table 4.1: Aggregated labels derived from majority voting, where w_1 and w_2 are independent workers and w_3, w_4 and w_5 are Sybil workers who occasionally deviate from the label sharing.

attacker may still identify a golden task when more than K Sybil workers are assigned to the task because a normal task is usually assigned to a fixed number K of workers. To hide the coordination among Sybil workers, the attacker may also allow them to occasionally deviate from the label sharing to make Sybil detection more difficult.

Example 6. *Suppose we have five tasks* $\{t_1, ..., t_5\}$ *labeled by two independent workers* $\{w_1,w_2\}$ *and three Sybil workers* $\{w_3, w_4, w_5\}$*, as shown in Table 4.1. The two independent workers provide the true label in most cases, while the three Sybil workers share a randomized but probably wrong label on each task with occasional deviation from the sharing. If majority voting is used for aggregating worker labels, i.e., assigning the same weight to each worker's labels, the aggregation result could be dominated by Sybil workers with a low aggregation accuracy. In this example, only* 40% *of aggregated labels match the true labels.* □

The main drawback of majority voting is that each worker is equally weighted when aggregating their labels. With the principle that the label provided by high-quality workers should be weighted more in the label aggregation, and the workers who provide labels more accurately should be assigned higher weights, *truth discovery* [54, 109] iteratively estimates the weight of workers and the true label of tasks by applying the EM algorithm [20]. However, online workers are still assumed to be independent and honest, which makes the method vulnerable to Sybil attack. For example, without prior knowledge of worker quality, the same weight is usually initialized for each worker, which may cause the same problem as majority voting.

Contributions. We make the following contributions toward defending strategic Sybil attacks in crowdsorucing:

1. We formalize the attack model of strategic Sybil attacks (Section 4.3) where the attacker attempts to evade detection through the identification of golden tasks and deviation of label sharing.

2. We propose an *extended truth discovery* (Section 4.4) to consider workers' Sybil behavior and labeling reliability in truth discovery using golden tasks.

3. We apply a *probabilistic task assignment* to strategically assign golden tasks while camouflaging them from the attacker.

4. To address the shortage of golden tasks caused by our camouflage, we design an online framework called Truth Discovery against Strategic Sybil Attack (*TDSSA*) (Section 4.6), where the *extended truth discovery* and *probabilistic task assignment* are performed in a batch mode to periodically promote tasks completed by high-quality workers as new golden tasks.

5. We conduct experiments (Section 4.7) using real and synthetic datasets with attack injection recommended in the literature to show that *TDSSA* can achieve a much higher aggregation accuracy than baseline methods under various attack settings and data characteristics.

4.2 Related Work

Improving the aggregation accuracy is one of the most studied topics in crowdsourcing. Towards this goal, many algorithms have been proposed, such as majority voting [64], truth discovery [54, 109] and probabilistic graphical models [98, 62] derived from the EM algorithm [20], and modeling the heterogeneity of worker quality [26, 61], worker community [85], task difficulty [112, 96], and task hierarchy [12, 24]. While these works advanced the state-of-the-art substantially, they all assume that workers independently provide their labels, which does not hold in the presence of Sybil attack.

To our knowledge, *SADU* [102] is the first work addressing Sybil attack in a general crowdsourcing setting. The idea is to cluster workers based on their label similarity and then classifies each cluster as Sybil or non-Sybil by assigning the same set of golden tasks with known true label to each worker in a cluster. As the clustering requires each pair of workers to label an adequate amount of same tasks, the separation of the two types of workers can be weakened by insufficient common tasks between workers. Also, when dealing with strategic Sybil attack, the method may assign a golden task to more than K Sybil workers in some cluster, where K is the number of workers hired for each normal task. Then, the identity of the golden task may be exposed, and the attacker could adaptively provide an honest label for Sybil workers to share, leading to the relatively low precision and recall of Sybil detection. Note that limiting each golden task to at most K workers requires restricting the cluster size to no more than K, which will affect the clustering function of *SADU*.

Copy detection [21, 55, 106] considers integrating data from many data sources on the Web where some sources may copy data from others. To explore an inter-dependence between source dependency and true values, and learn the two iteratively, these methods rely on most of the aggregated values in the first iteration to be correct. We may treat Sybil

workers as copiers from the same source for Sybil detection. However, without the ground truth of tasks, this initial dependency cannot distinguish the label sharing of Sybil workers from the label convergence of high-quality independent workers. Applying golden tasks in a cold start phase may not help because a strategic Sybil attacker can identify such golden tasks.

Other attacks that could subvert a crowdsourcing system were also studied in the literature. In a spam attack [39], a spammer may provide randomized or even malicious labels. Since spammers do not share labels, it is hard for them to dominate the aggregation result. Data poisoning attack [65, 66, 27] aims to maximize the number of tasks whose aggregated labels are different from the true ones, by assuming that the attacker has full knowledge about other workers' labels on the same task. Unlike these attacks, the goal of Sybil attack is to get rewards with minimum effort and is less concerned with whether the provided label differs from the true label.

In online social networks (OSNs), a Sybil attacker may coordinate "Sybil users" to unfairly increase the reputation of targeted users or items. Detecting such users in OSNs often relies on analyzing their social structures or other features. In some specific crowdsourcing applications such as mobile mapping service [90, 91] and fake-review detection [108, 43], this technique could be applied. However, in most cases of crowdscouring, the interconnection and features of dynamic online workers are not available.

4.3 Problem Definition

We consider a set of N tasks, denoted by $T = \{t\}$, and a set of M workers, denoted by $W = \{w\}$. Each task has L optional labels, only one of which is the *true* label. Each worker can request for one task at a time and must provide a label once a task is assigned. To *complete* a task, K workers are hired to provide their labels on the task, where K is usually a small constant determined by the budget. Let $l_{t,w}$ be a worker w's label on a task t. The true labels are approximated by the aggregated labels $\mathcal{L}^a = \{l_t^a\}$ derived from the worker label matrix $\mathcal{L} = [l_{t,w}]$. The *aggregation accuracy* is the probability for the aggregated label to be identical to the true label. A small set of *golden tasks* $T' = \{t'\}$ with known true labels $\mathcal{L}' = \{l'_t\}$ can be used to test worker quality. Table 4.2 summarizes the frequently used notations, referred as the $GLOBAL$ variables used by our algorithms.

Problem 2. *Given a normal task set T and a (small) golden task set T', we aim to achieve a high aggregation accuracy for the tasks in T by assigning tasks in T and T' to requesting workers from an online worker set W, where W may contain Sybil workers controlled by strategic Sybil attackers.*

Attack Model. We assume that a strategic Sybil attacker may coordinate several Sybil worker accounts to share a randomized label on each task. We stress that the attacker's goal

Table 4.2: The set of global variables referred as *GLOBAL* for TDSSA

Symbol	Description
$T = \{t\}$	N normal tasks with unknown true labels
$W = \{w\}$	M online workers (may contain Sybil workers)
$T' = \{t'\}$	golden tasks with known true labels
$\mathcal{L} = [l_{t,w}]$	a matrix of worker label $l_{t,w}$ for each pair of worker w and task t
$\mathcal{L}^a = \{l_i^a\}$	aggregated label of completed tasks
$\mathcal{L}^* = \{l_{t'}^*\}$	true label of golden tasks in T'
$S = \{s_w\}$	Sybil score that measures workers' Sybil behavior
$R = \{r_w\}$	reliability score that indicates workers' labeling reliability
$A = \{a_w\}$	estimated accuracy of workers
K	the number of workers hired for each normal task in T
L	the number of optional labels for a task
τ	Sybil threshold for banning workers
δ	reliability threshold for marking reliable workers
α	probability to assign a golden task to a new worker
B	condition for terminating a batch

is not to mislead the aggregation to a wrong result, but to make easy money by dominating the aggregation without knowing independent workers' labels. Therefore, the solutions to many other attacks discussed in Section 4.2 cannot be applied to solve Problem 2, and our *TDSSA* framework is not designed for other attacks either. The public knowledge on K and the assignment of golden tasks would be utilized by the attacker to evade detection. For example, golden tasks could be identified if they are assigned in a cold-start phase or, in the case of random assignment, assigned to more than K Sybil workers under the attacker's control. The attacker may also allow each Sybil worker to occasionally deviate from the sharing behavior by providing a randomized label different from the shared one.

4.4 Extended Truth Discovery

In this section, we first show the vulnerability of truth discovery [54, 109] under Sybil attack (Section 4.4.1). Then, we apply golden tasks to associate each worker with two parameters, called Sybil score and reliability score, which would reveal the worker's Sybil behavior and labeling quality (Section 4.4.2). Finally, we extend truth discovery using these two parameters to suppress the impact of Sybil workers and low-quality independent workers (Section 4.4.3).

4.4.1 Truth Discovery

Truth discovery is a learning algorithm that iteratively estimates the weight of workers and the true label of tasks. Let W_t denote the set of workers on task t, T_w denote the set of tasks labeled by worker w, and $\mathbb{1}(x, y)$ denote the indicator function that returns 1 if $x = y$

or 0 otherwise. Truth discovery conducts the following two steps to estimate the aggregated label $l_t^a(i)$ of each task t and update the weight $h_w(i)$ of each worker w at each iteration i until convergence:

Label Aggregation updates the aggregated label $l_t^a(i)$ for each task t based on the weight $h_w(i-1)$ of each worker w estimated in the previous iteration $i-1$:

$$l_t^a(i) = \operatorname*{argmax}_l \sum\nolimits_{w \in W_t} h_w(i-1) \cdot \mathbb{1}(l_{t,w}, l). \tag{4.1}$$

In words, the aggregated label $l_t^a(i)$ is the label that receives the highest total weight from workers in W_t who label task t, which allows workers with a higher weight to contribute more in the aggregation.

Weight Estimation updates the weight $h_w(i)$ for each worker w based on how often w agrees with the aggregated label:

$$h_w(i) = \frac{\sum_{t \in T} \mathbb{1}(l_{t,w}, l_t^a(i))}{|T_w|}. \tag{4.2}$$

By treating the aggregated label $l_t^a(i)$ as the true label, $h_w(i)$ is worker w's estimated accuracy on the tasks in T_w.

Without prior knowledge of worker quality, the weight of each worker is usually initialized to 1 before the first iteration. The next example shows that such a uniform initialization of worker weight may cause the same problem as majority voting in the presence of Sybil attack.

Example 7. *Recall the five workers in Table 4.1. With $h_w(0) = 1$ of each worker w, Equation 4.1 would produce the same aggregation result as majority voting. Subsequently, based on Equation 4.2, we have $h_{w_1}(1) = 0.4$, $h_{w_2}(1) = 0.2$, $h_{w_3}(1) = 1.0$, $h_{w_4}(1) = 0.8$ and $h_{w_5}(1) = 0.6$, so Sybil workers would exert more influence than independent workers in the next iteration.* □

4.4.2 Sybil Score and Reliability Score

The vulnerability of truth discovery under Sybil attack mainly comes from the assumption that workers may have diverse expertise but label tasks independently. Since Sybil workers share labels randomized by the attacker, the chance for them to provide a *false majority* label on a golden task would be high. Based on this intuition, we evaluate a Sybil score for each worker using golden tasks.

Sybil Score (s_w) captures a worker w's Sybil behavior of providing a false majority label on golden tasks, defined as

$$s_w = \frac{2}{1 + e^{-\sum_{t' \in T'} \mathbb{1}(l_{t',w}, l_{t'}^m) \cdot (1 - \mathbb{1}(l_{t',w}, l_{t'}^*))}} - 1, \tag{4.3}$$

where $l^*_{t'}$ is the known true label of a golden task t' and $l^m_{t'}$ is a label voted by the majority of workers on t'. $\mathbb{1}(l_{t',w}, l^m_{t'}) \cdot (1 - \mathbb{1}(l_{t',w}, l^*_{t'})) = 1$ if and only if $l_{t',w}$ is a majority but false label, which is an evidence of sharing a randomized label. By rescaling the logistic sigmoid function, s_w ranges between 0 and 1 and monotonically increases with more of such evidences.

Unlike the cold-start phase of requiring each new worker to label the same golden tasks in our *GAA*/*IGAA* algorithms (Chapter 3), we intend to strategically distribute golden tasks for camouflaging them from the attacker. In this case, our trust on a worker w cannot be simply measured by the worker's accuracy p_w, computed by

$$p_w = \begin{cases} \frac{1}{L} & \text{if } |T'_w| = 0 \\ \frac{\sum_{t' \in T'_w} \mathbb{1}(l_{t',w}, l^*_{t'})}{|T'_w|} & \text{otherwise} \end{cases} \tag{4.4}$$

where $|T'_w|$ may vary for different workers. For example, a worker w who has correctly labeled a single golden task will have 100% accuracy, but we should trust another worker w' more if the worker has labeled 10 golden tasks with 90% accuracy. Therefore, we evaluate a reliability score for each worker by considering how many golden tasks the worker has labeled.

Reliability Score (r_w) measures the labeling reliability of a worker w using golden tasks, defined as

$$r_w = (\frac{2}{1 + e^{-|T'_w|}} - 1) \cdot p_w, \tag{4.5}$$

where the first multiplicative term factors in the number of golden tasks T'_w labeled by w. r_w ranges between 0 and 1 and would be high only if worker w has labeled several golden tasks with a high accuracy. This requirement is more than having a low Sybil score because the latter could come from providing false but non-majority labels, which fails to ensure a high reliability score. However, a high Sybil score usually implies a low accuracy on golden tasks, and thus, a low reliability score.

Given a *Sybil threshold* τ and a *reliability threshold* δ, each worker w can be marked as one of "banned" ($s_w \geq \tau$), "reliable" ($r_w \geq \delta$), or "undetermined" ($s_w < \tau$ and $r_w < \delta$). Initially, a worker w not tested by any golden task is "undetermined" with $s_w = r_w = 0$. As being tested by more golden tasks, workers may become "banned" or "reliable". "Reliable" workers are trusted and will not be tested by more golden tasks, while "banned" workers will have their labels on normal tasks removed to improve the aggregation accuracy and are not allowed to request for more tasks. However, "banned" workers' labels on golden tasks will be kept for detecting more Sybil workers.

Example 8. *A larger Sybil threshold τ enhances our confidence to ban a worker. With $\tau = 0.7$, a worker w who mistakenly provides a false majority label on 2 golden tasks has $s_w = 0.76$ and thus is "banned". If τ is increased to 0.8, worker w will be banned after*

Algorithm 4: Extended Truth Discovery ETD

Global: $GLOBAL$

Output: $\mathcal{L}^a$: aggregated label of tasks

```
Set h_w(0) to p_w for each worker w ∈ W;
i = 1;
while L^a is not converged do
    for each task t ∈ T do
        Update the aggregated label l_t^a(i) by Equation 4.6;
    for each worker w ∈ W do
        Update the worker weight h_w(i) by Equation 4.7;
    i = i + 1;
Return L^a;
```

providing a false majority label on 3 golden tasks. A larger optional label size L may increase the Sybil score of Sybil workers because the randomized label shared by these workers on a golden task have a higher chance to be a false label. A larger reliability threshold δ enhances our confidence to trust a worker. With $\delta = 0.7$, a worker w who provides 3 true labels on 4 golden tasks (i.e., $|T'_w| = 4$) has $r_w = 0.72$ and thus is marked as "reliable". If δ is increased to 0.8, worker w is "reliable" only if all the 4 golden tasks are correctly labeled. □

4.4.3 Extending Truth Discovery by s_w and r_w

Our extended truth discovery (ETD) in Algorithm 4 integrates the Sybil scores $S = \{s_w\}$ and reliability scores $R = \{r_w\}$ into truth discovery, as described below.

Extended Label Aggregation extends Equation 4.1 to update the aggregated label $l_t^a(i)$ for each task t by

$$l_t^a(i) = \arg\max_{l} \sum_{w \in W_t} [(1 - s_w) \cdot h_w(i-1) + s_w \cdot \frac{1}{L}] \cdot \mathbb{1}(l_{t,w}, l), \tag{4.6}$$

where $h_w(0) = p_w$. In other words, each worker $w \in W_t$ has $1 - s_w$ probability of being an independent worker with $h_w(i-1)$ accuracy, and s_w probability of being a Sybil worker with $\frac{1}{L}$ accuracy. A large Sybil score renders the weight of a worker to the random guess $\frac{1}{L}$, whereas a small Sybil score renders the label aggregation to the standard one. In particular, Equation 4.1 is the special case of Equation 4.6 with $s_w = 0$ for each worker w.

Extended Weight Estimation extends Equation 4.2 to update the weight $h_w(i)$ for each worker w by

$$h_w(i) = \frac{\sum_{t \in T_w} c_t \cdot \mathbb{1}(l_{t,w}, l_t^a(i))}{\sum_{t \in T_w} c_t}, \tag{4.7}$$

where c_t is the average reliability score of workers W_t who label a task t, computed by

$$c_t = \frac{\sum_{w \in W_t} r_w}{|W_t|}. \tag{4.8}$$

We weigh each task t by the average reliability score of workers on the task. Note that the reliability score r_w and the weight $h_w(i)$ serve different purposes: the former is the trust on worker w gained by labeling golden tasks, and the latter is the accuracy of worker w estimated based on the aggregated labels. Equation 4.2 is the special case of Equation 4.7 with $r_w = 1$ for each worker w.

Example 9. *Let's reconsider Example 7, assuming that the Sybil workers w_3, w_4 and w_5 provide the true label on 3 golden tasks and share a false label on 2 golden tasks (thus, $p_{w_3} = p_{w_4} = p_{w_5} = 0.6$), and the independent workers w_1 and w_2 label all the 5 golden tasks correctly (thus, $p_{w_1} = p_{w_2} = 1$). From Equation 4.3, $s_{w_1} = s_{w_2} = 0$ and $s_{w_3} = s_{w_4} = s_{w_5} = 0.76$. Then, based on Equation 4.6, the total weight of w_1 and w_2 is 2, while the total weight of the 3 Sybil workers would be lowered to $3 \cdot (0.76 \cdot \frac{1}{3} + (1 - 0.76) \cdot 0.6) = 1.185$. Sybil workers' weight is correctly suppressed in label aggregation.* □

In each iteration of Algorithm 4, the extended label aggregation runs in $O(N)$ time because each task is labeled by a constant number K of workers. The weight estimation updates the weight of M workers, but each worker labels $\frac{K \cdot N}{M}$ tasks on average, so the running time is also $O(N)$. The iteration number is constant if the iterative learning terminates when the change of the aggregated labels is smaller than a pre-defined threshold [56], e.g., 1% in our experiment. Therefore, the complexity of Algorithm 4 is $O(N)$.

4.5 Probabilistic Task Assignment

We now present our probabilistic task assignment with the camouflage of golden tasks.

4.5.1 Assigning Tasks to Requesting Workers

The design of our online task assignment is motivated by strategically assigning golden tasks while camouflaging them from the attacker. To achieve this, we *probabilistically* assign a golden task $t' \in T'$ or a normal task $t \in T$ to each requesting worker w based on the worker's Sybil score s_w and reliability score r_w. A "banned" worker cannot request for more tasks and a "reliable" worker would not be tested by more golden tasks (see Section 4.4.2), so golden tasks are only assigned to "undetermined" workers.

Our probabilistic task assignment PTA is shown in Algorithm 5. If a requesting worker w is in the "undetermined" status, we assign a valid golden task with $g_w(\alpha)$ probability (Step 2) defined by

$$g_w(\alpha) = \alpha \cdot (1 - r_w) + (1 - \alpha) \cdot s_w. \tag{4.9}$$

Algorithm 5: Probabilistic Task Assignment $PTA(w)$

Input: w: a requesting worker
Global: $GLOBAL$

```
1 if worker w is in the "undetermined" status (s_w < τ and r_w < δ) then
2     Assign worker w to a valid golden task t' ∈ T' with g_w(α) probability computed
        by Equation 4.9;
3     if worker w is not assigned to any golden task then
4         Assign worker w to an uncompleted normal task t ∈ T;
5 else
6     Assign worker w to an uncompleted normal task t ∈ T;
```

This ensures that a worker with a higher Sybil score and/or a lower reliability score is more likely to be tested by golden tasks. $\alpha \in (0, 1]$ is the probability for assigning a golden task to a new worker w with s_w and r_w both initialized to 0. α also serves as a trade-off for the influence of s_w and r_w on $g_w(\alpha)$. If no golden task is assigned or if worker w is in the "reliable" status, we assign an uncompleted normal task $t \in T$ (Step 3-6). The restriction for assigning a valid golden task (Step 2) will be discussed in Section 4.5.2. Algorithm 5 can be run in $O(1)$ time by tracking the validity of golden tasks in T'.

In the following, we analyze the risk of exposing a golden task to the attacker by $g_w(\alpha)$, assuming the attacker has the knowledge about the task assignment algorithm, the parameter α, and the thresholds τ and δ. First, learning $g_w(\alpha)$ in Equation 4.9 requires knowing the values of r_w and s_w, thus, identifying the type of each task assigned, which goes back to learning $g_w(\alpha)$. Hence, the attacker does not learn the exact value of $g_w(\alpha)$. However, since a golden task is assigned only to an "undetermined" worker w with $s_w < \tau$ and $r_w < \delta$, we have $\alpha \cdot (1 - \delta) + (1 - \alpha) \cdot s_w < g_w(\alpha) < \alpha \cdot (1 - r_w) + (1 - \alpha) \cdot \tau$. For example, with $\alpha = 0.5$ and $\delta = \tau = 0.8$, we have $0.1 + 0.5 \cdot s_w < g_w(\alpha) < 0.9 - 0.5 \cdot r_w$. For a Sybil worker w that is not "banned", we have $s_w < 0.8$, $r_w \geq 0$ and $0.5 < g_w(\alpha) < 0.9$. Therefore, the attacker learns that an assigned task is a golden task with a probability in the interval $(0.5, 0.9)$, but the exact probability is unknown. This disclosure is expected because we want to test more frequently a worker with a large s_w and a small r_w. While the equal probability $g_w(\alpha) = 0.5$ provides the maximum uncertainty for learning the task type, it could leave insufficient golden tasks for workers that need more tests because of the restriction on valid golden tasks discussed below.

4.5.2 Risk Analysis of "Reliable" Workers

In Step 2 of Algorithm 5, a golden task is valid if it has been previously assigned to less than K workers with the "banned" or "undetermined" status. As these workers are potential Sybil workers, the restriction prevents a golden task from being identified by a strategic Sybil attacker. Note that "reliable" workers ate not counted in the restriction, so there is a risk

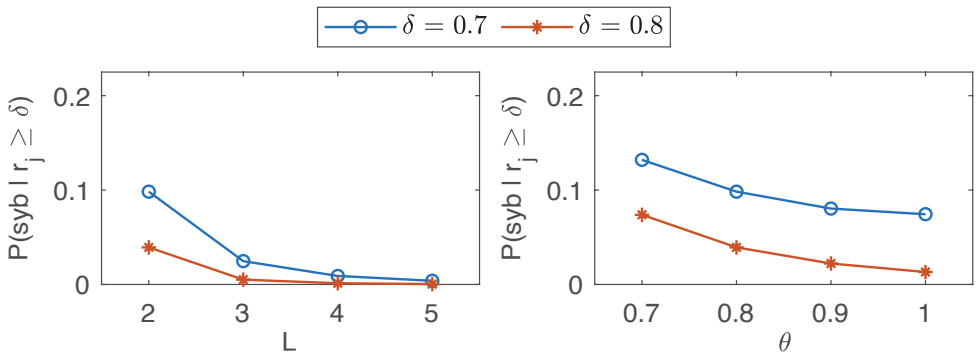

Figure 4.1: $P(syb|r_w \geq \delta)$ vs L and θ, with $\mu = 0.3$, $|T'_w| = 5$, $L = 2$ and $\theta = 0.8$ by default.

for a golden task to be assigned to more than K Sybil workers if a "reliable" worker w (i.e., $r_w \geq \delta$) on the task is indeed a Sybil worker. Let $P(syb|r_w \geq \delta)$ denote the probability that a "reliable" worker w is actually a Sybil worker. Below we estimate $P(syb|r_w \geq \delta)$.

Let $P(r_w \geq \delta)$ be the probability for a worker w to pass the reliability threshold δ, $P(r_w \geq \delta|syb)$ be the probability to have $r_w \geq \delta$ given w is a Sybil worker, and $P(r_w \geq \delta|ind)$ be the probability to have $r_w \geq \delta$ given w is an independent worker. If the proportion of Sybil workers in W is μ, we have

$$P(syb|r_j \geq \delta) = \frac{\mu \cdot P(r_w \geq \delta|syb)}{\mu \cdot P(r_w \geq \delta|syb) + (1-\mu) \cdot P(r_w \geq \delta|ind)}. \tag{4.10}$$

By Equation 4.5, a worker w is "reliable" only if w provides the true label on at least $d = \lceil \delta \cdot |T'_w| / (\frac{2}{1+e^{-|T'_w|}} - 1) \rceil$ golden tasks because

$$r_w = (\frac{2}{1+e^{-|T'_w|}} - 1) \cdot p_w \geq \delta \Rightarrow p_w \geq \frac{\delta}{\frac{2}{1+e^{-|T'_w|}} - 1}. \tag{4.11}$$

Then $P(r_w \geq \delta)$ is equal to the probability of providing at least d true labels on $|T'_w|$ golden tasks. Let θ_w be the probability for a worker w to provide the true label on any task, which is actually a Bernoulli trial with the success probability θ_w. So we have

$$P(r_w \geq \delta) = \sum_{x=d}^{|T'_w|} \binom{|T'_w|}{x} \cdot (\theta_w)^x \cdot (1-\theta_w)^{|T'_w|-x} \tag{4.12}$$

Theorem 3. *Given the optional label size L and the average accuracy θ of independent workers, $P(syb|r_w \geq \delta)$ is computed by Equation 4.10, where $P(r_w \geq \delta|syb)$ is $P(r_w \geq \delta)$ with $\theta_w = \frac{1}{L}$ and $P(r_w \geq \delta|ind)$ is $P(r_w \geq \delta)$ with $\theta_w = \theta$.*

Assume the Sybil proportion μ is 0.3 and each worker w labels $|T'_w| = 5$ golden tasks. Figure 4.1 shows $P(syb|r_w \geq \delta)$ vs the optional label size L and the average worker accuracy θ for different δ. As L or θ increases, fewer Sybil workers or more independent workers are

Algorithm 6: *TDSSA*

Global: *GLOBAL*
Output: $\mathcal{L}^a$: aggregated label of tasks

```
1  while not all tasks in T are completed do
2      while batch condition B is not met do
3          switch a worker w ∈ W requests or labels a task do
4              case 1. w sends a request do
5                  Assign w to a task by PTA(w);
6              case 2. w labels a golden task t' ∈ T' do
7                  Update s_w, r_w, p_w (Equation 4.3, 4.5, 4.4);
8                  Update c_t for all tasks t labeled by w (Equation 4.8);
9                  if s_w passes the Sybil threshold τ then
10                     ban w with her labels on normal tasks in T removed from L;
11             case 3. w submits a label l_{t,w} on a normal task t ∈ T do
12                 Add the label l_{t,w} to L;
13     Update L^a by ETD;
14     Promote completed tasks t ∈ T \ T' with c_t ≥ δ to T';
15 Return L^a;
```

"reliable", so $P(syb|r_w \geq \delta)$ decreases quickly. A larger δ requires a worker w to provide more true labels on $|T'_w|$ golden tasks in order to become "reliable". This is more difficult for Sybil workers than for independent workers because the former has only $\frac{1}{L}$ accuracy, therefore, $P(syb|r_w \geq \delta)$ drops. The HIT requester can use δ to control the risk $P(syb|r_w \geq \delta)$.

4.6 *TDSSA* Framework

Now we present the overall *TDSSA* as Algorithm 6. The idea is to run *ETD* and *PTA* in a batch mode to periodically update the worker parameters, i.e., $S = \{s_w\}, R = \{r_w\}, P = \{p_w\}$, and promote completed tasks as new golden tasks.

TDSSA responds to each action of a worker w differently. A task will be assigned by $PTA(w)$ when w makes a request (case 1). If w labels a golden task in T' (case 2), the worker parameters will be updated, and w may be banned with her labels on normal tasks in T removed once s_w passes the Sybil threshold τ. If w labels a normal task in T (case 3), the worker label matrix $\mathcal{L}$ will be updated. Upon the satisfaction of the batch condition B (Step 2), *ETD* updates $\mathcal{L}^a$ based on collected worker labels $\mathcal{L}$, and the completed tasks $t \in T \setminus T'$ with $c_t \geq \delta$ are promoted as new golden tasks (Step 12-13), where the aggregated label of a promoted task will be treated as the true label and will not be further updated. A promoted task will not be used to test any worker who completed the task because the related label is already known.

The batch condition B serves as a trade-off between the efficiency of *TDSSA* and the freshness of golden tasks T' and worker quality $\{S, R, P\}$. For example, B can be specified in terms of the number of "promotable" tasks in T, which affects the changes of T' and $\{S, R, P\}$. If most workers are high-quality independent workers, many completed tasks could be promoted to continuously update $\{S, R, P\}$. If many workers are Sybil workers or low-quality independent workers, valid golden tasks may be run out with few promotion and $\{S, R, P\}$ cannot be further updated. However, the $\{S, R, P\}$ obtained in early batches will continue suppressing the impact of Sybil and low-quality workers in subsequent batches.

The complexity of *TDSSA* mainly comes from ETD (Algorithm 4) and PTA (Algorithm 5). Since the number of batches is bounded by the task number N and Algorithm 4 converging in $O(N)$ time in each batch, the total running time of ETD is $O(N^2)$. The total running time of PTA is $O(KN)$ because exactly K workers are hired for each of the N tasks in T and Algorithm 5 takes $O(1)$ time to assign a task to a requesting worker. Therefore, the complexity of *TDSSA* is $O(N^2)$. When dealing with a large N, we may terminate a batch based on the percentage of tasks in T being completed, which would lead to a constant number of batches and drop the complexity of *TDSSA* to $O(N)$.

4.7 Experiment

This section evaluates *TDSSA* in the presence of strategic Sybil attack. On a real crowdsoucing platform, strategic Sybil attack is not guaranteed to be present in a particular experiment and, even when present, it is hard to know which workers are actually Sybil workers for evaluating the performance. So we used previously collected crowdsourcing data and adopted the Sybil injection recommended by *SADU* [102]. All experiments were implemented in Java on a computer with CPU 3.60GHz and 32GB memory.

4.7.1 Experiment Settings

Real Datasets. We used two public real crowdsourcing datasets, named NLP [110] and DOG [112, 102], both of which were collected from AMT. The NLP dataset was used in a crowdsourcing survey, where $M = 85$ workers label $N = 1000$ tweets (tasks) as positive or negative sentiments and each tweet is labeled by $K = 20$ workers. The DOG dataset was used to evaluate several crowdsourcing models, such as *D&S* [112] and *SADU* [102], where $M = 109$ workers label $N = 807$ dog images (tasks) as one of four types and each image is labeled by $K = 10$ workers. Both datasets are represented by a set of $(t, w, l_{t,w})$ tuples, indicating worker w provides label $l_{t,w}$ on task t. The true labels of all tasks are also given by these two datasets.

Synthetic Datasets. As data properties are fixed for real datasets, we also created several synthetic datasets, denoted by SYN, in order to test the performance of different algorithms under various task number N, worker number M, worker number K per task,

Table 4.3: Data characteristics

Properties	NLP	DOG	SYN (default value)
Task Number N	1000	807	1000∼9000 (5000)
Worker Number M	85	109	100∼900 (500)
Workers per Task K	20	10	5∼20 (10)
Optional Label Size L	2	4	2∼5 (4)
Worker Accuracy θ	0.8	0.7	0.7∼1 (0.8)

Table 4.4: Attack characteristics

Symbol	**Description**	**Settings** (default value)
μ	Percentage of Sybil workers	0∼0.8 (0.5)
ϵ	Probability for Sybil workers to deviate from sharing	0∼0.3 (0.1)
λ	Number of attackers	1∼4 (1)

optional label size L, and average worker accuracy θ. The $(t, w, l_{t,w})$ tuples were generated based on the average worker accuracy θ. For example, with $\theta = 0.8$, a worker will have 0.8 probability to provide the true label and 0.2 probability to provide a false label.

We summarize the statistics of real and synthetic datasets in Table 4.3. The default setting is used if no specification is given.

Golden Task Simulation. NLP comes with a pool T' of 20 golden tasks and the tuples $(t', w, l_{t',w})$ for each worker w and each golden task $t' \in T'$. For DOG and SYN, we created 20 golden tasks T' and used the worker accuracy of the original data to generate the tuples $(t', w, l_{t',w})$ for each worker w and each golden task $t' \in T'$, through the same procedure for creating the synthetic datasets.

Sybil Injection. We adopted the Sybil injection of *SADU* [102] using three parameters (μ, ϵ, λ). A proportion μ of independent workers were replaced with Sybil workers evenly distributed among λ attackers. On each task, the Sybil workers controlled by the same attacker will share a randomized label with probability $1 - \epsilon$ and independently randomize a label with probability ϵ. The attacker adaptively switches from randomization to an honest label once a golden task is identified. The settings of (μ, ϵ, λ) are summarized in Table 4.4.

Evaluation Metrics. We examined three performance metrics: (1) *Aggregation accuracy* (**A-accuracy**) is the percentage of tasks whose aggregated label computed by an algorithm is identical to the true label. (2) *Exposure number* (**E-number**) is the number of golden tasks assigned to more than K Sybil workers under the same attacker's control. The attacker will identify such golden tasks and will switch from randomization to an honest label (derived from the average worker accuracy) to evade detection. This behavior only

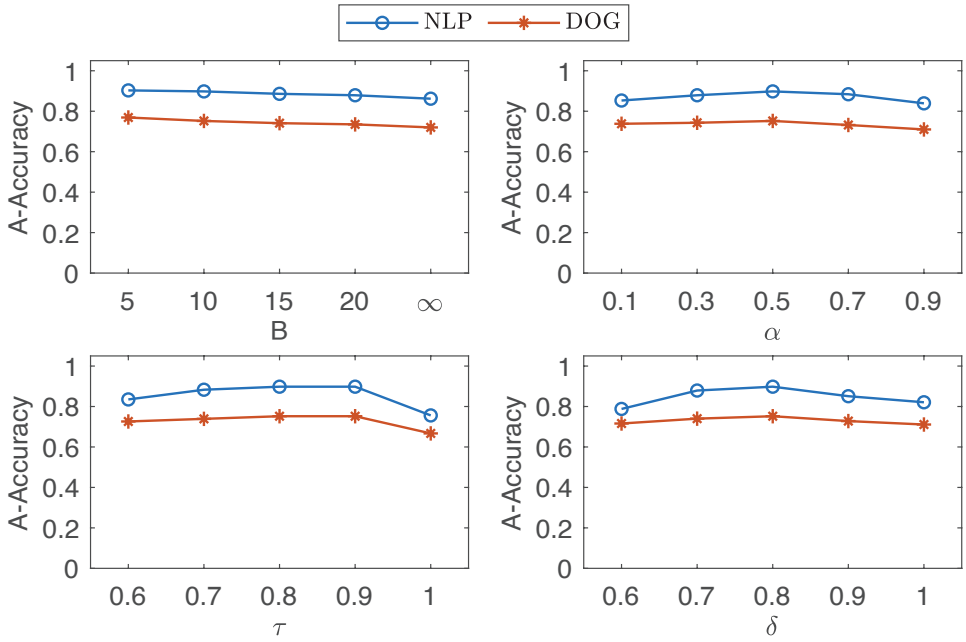

Figure 4.2: A-accuracy of *TDSSA* vs batch condition B, initial golden task assignment probability α, Sybil threshold τ and reliability threshold δ on NLP and DOG

affects algorithms that use golden tasks. (3) *Testing cost* (**T-cost**) is the average number of golden tasks assigned to each worker, representing the overhead due to the testing.

4.7.2 Settings of Parameters for *TDSSA*

Our *TDSSA* framework is parameterized by the batch condition B, the initial golden task assignment probability α, the Sybil threshold τ, and the reliability threshold δ. Figure 4.2 shows the A-accuracy of *TDSSA* vs these parameters on the NLP and DOG datasets under the default setting of attack parameters in Table 4.4.

We consider the batch condition B defined by the number of "promotable" tasks. A tighter B allows a slightly higher A-accuracy at a higher computational overhead for running the extended truth discovery. A single batch, indicated by $B = \infty$, only suffers from a small drop in A-accuracy. This shows that, even without promoting completed tasks into new golden tasks, the worker parameters obtained from the initial set of golden tasks can still improve the aggregation result.

The parameter α not only initially defines the probability of assigning golden tasks to a worker not tested by any golden task yet, but also acts as the trade-off between Sybil score and reliability score when we compute this probability later on. As we can see, *TDSSA* reaches the highest A-accuracy if we equally weight both scores.

The A-accuracy of *TDSSA* is reduced when the Sybil threshold τ is too small, e.g., $\tau = 0.6$, because independent workers may be mistakenly banned. If τ is too high, e.g., $\tau = 1$, no Sybil worker would be banned, and the A-accuracy also drops. For the reliability

threshold δ, there is a similar trend of reduction in the A-accuracy, where more Sybil workers could be mistakenly marked as "reliable" with a too small δ or few completed tasks can be promoted with a too large δ.

In our experiments, we choose $B = 10$, $\alpha = 0.5$, $\tau = 0.8$, $\delta = 0.8$ for *TDSSA*.

Competing Algorithms.

We compare *TDSSA* with the following three baseline methods.

1. *TD* is the standard truth discovery method that iteratively estimates the weight of workers and the true label of tasks.

2. *TD-DEP* is the ACCU model for copy detection [21]. By applying this method to strategic Sybil attack, the label sharing of Sybil workers is treated as copying labels from the same source. The same parameter settings as specified in the paper were applied.

3. *SADU* [102] is a clustering-based algorithm that groups workers based on their label similarity and uses golden tasks to detect the groups mainly containing Sybil workers. We applied the parameter settings and used a subset of 10 golden tasks for testing workers in a cluster, as recommended by the paper.

For truth discovery in *TD*, *TD-DEP* and our *TDSSA*, we set a convergence threshold of 0.001, i.e., the iterative learning terminates when the change in aggregated answers, compared to those in the previous iteration, is less than 0.1%. We also set the maximum number of iterations to 1000 to prevent non-convergence, but we found that all these methods quickly converge with a few iterations in our experiments.

Each algorithm is run for 50 times with the averaged result reported. In each run, we apply a randomized sequence of worker requests with a randomized Sybil injection. To preserve the labels given by the original datasets, a worker w is assigned to a task t only if the dataset contains the tuple $(t, w, l_{t,w})$. Sybil workers' labels on an assigned task were generated on the fly as described above.

4.7.3 Experiment on Real Datasets

Figure 4.3 shows A-accuracy (left) and E-number (right) vs varying attack parameters (μ, ϵ, λ) on the NLP dataset.

A-Accuracy. *TDSSA* has the *highest* A-accuracy across all tested μ, ϵ and λ. The A-accuracy of *TDSSA* drops only slightly when the Sybil proportion μ increases up to 0.6, thanks to suppressing the impact of Sybil workers and low-quality independent workers and camouflaging golden tasks. *TD* and *SADU* have a much lower A-accuracy than *TDSSA* for a larger μ. For *SADU*, this is because the attacker evades the testing of the golden tasks that were identified by the attacker. See the discussion on E-number below. *TD-DEP*

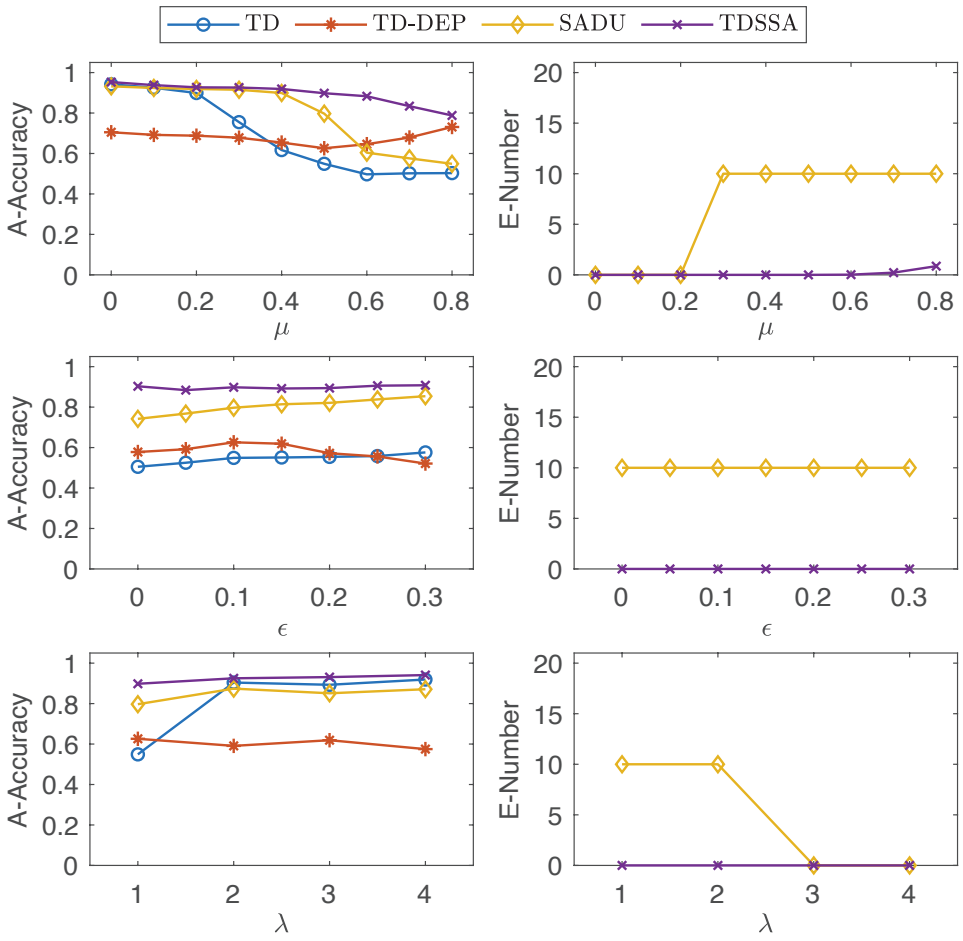

Figure 4.3: A-accuracy and E-number vs attack parameters (μ, ϵ, λ) on the NLP dataset. We vary one parameter at a time with $\mu = 0.5$, $\epsilon = 0.1$ and $\lambda = 1$ by default.

has a lower A-accuracy than *TD* on NLP for a small μ. Without using golden tasks, the dependency analysis of *TD-DEP* failed to distinguish the sharing behavior of Sybil workers and the label convergence of high-quality independent workers.

A larger ϵ weakens the Sybil behavior of sharing labels and leads to an increase in A-accuracy for all algorithms because Sybil workers have a lower chance to dominate the aggregation result. The A-accuracy of *TD-DEP* starts to drop when $\epsilon \geq 0.15$ where it is hard to distinguish the weakened dependency of Sybil workers and the label convergence of high-quality independent workers.

With the default $\mu = 0.5$, a larger attacker number λ implies that each attacker controls fewer Sybil workers, making it harder to dominate the aggregation result. For *TD-DEP*, this means it is harder to detect the dependency between Sybil workers, therefore, the A-accuracy of *TD-DEP* does not improve like *TD*.

E-Number. A higher E-number indicates more exposures of golden tasks to the attacker. For *TDSSA*, a golden task may be exposed when some "reliable" workers are actually

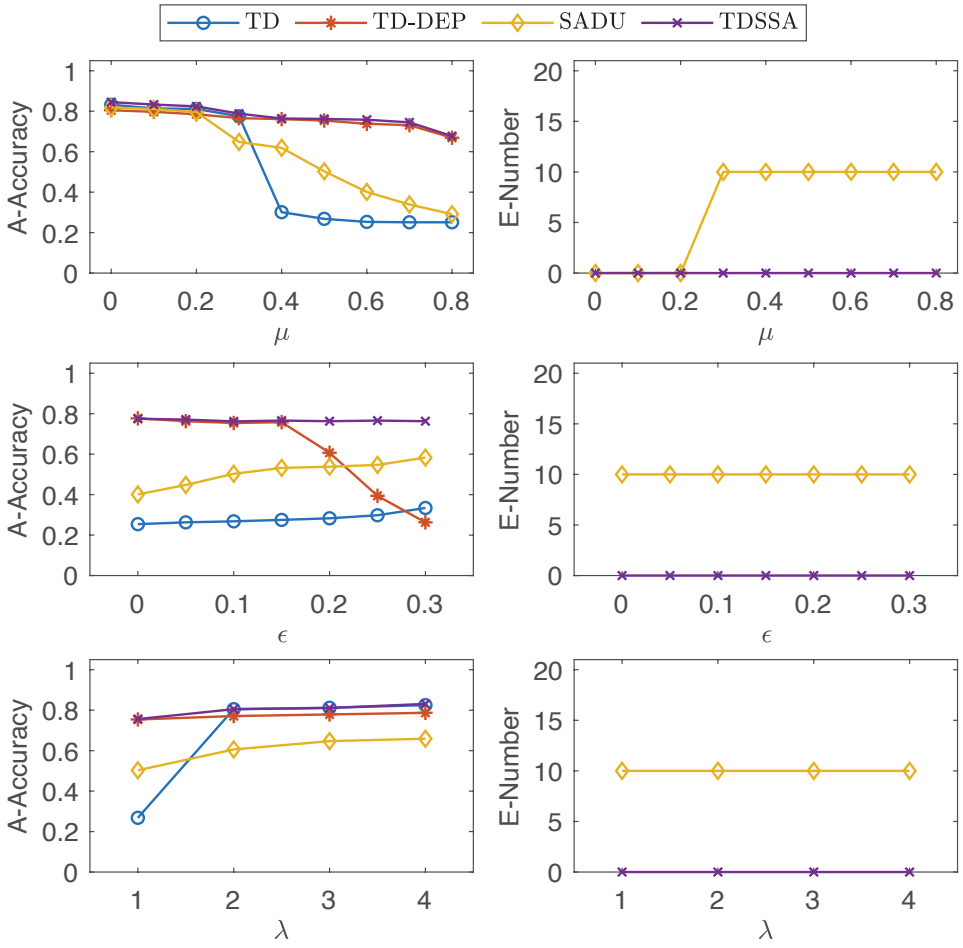

Figure 4.4: A-accuracy and E-number vs attack parameters (μ, ϵ, λ) on the DOG dataset. We vary one parameter at a time with $\mu = 0.5$, $\epsilon = 0.1$ and $\lambda = 1$ by default.

Sybil workers (see Section 4.5.2). This was observed for a large μ for NLP where the small optional label size $L = 2$ makes it easier to mark a Sybil worker as a "reliable" worker. For DOG with $L = 4$, the E-number of *TDSSA* remains 0. For many settings of *SADU*, one cluster contains more than K Sybil workers controlled by the same attacker, so the 10 golden tasks assigned to the cluster were identified, therefore, the E-number is 10. With the high E-number, we observed only precision and recall between 60% and 80% on our real datasets, which contribute to the low A-accuracy of *SADU*.

T-Cost. *SADU* has a fixed T-cost of 10 because each worker is tested by 10 golden tasks. The T-cost of *TDSSA* is 5.05 (with 0.19 variance) for NLP and 1.83 (with 0.08 variance) for DOG. The lower T-cost benefits from *TDSSA*'s strategic assignment of golden tasks based on Sybil score and reliability score.

Figure 4.4 shows A-accuracy (left) and E-number (right) vs varying attack parameters (μ, ϵ, λ) on the DOG dataset.

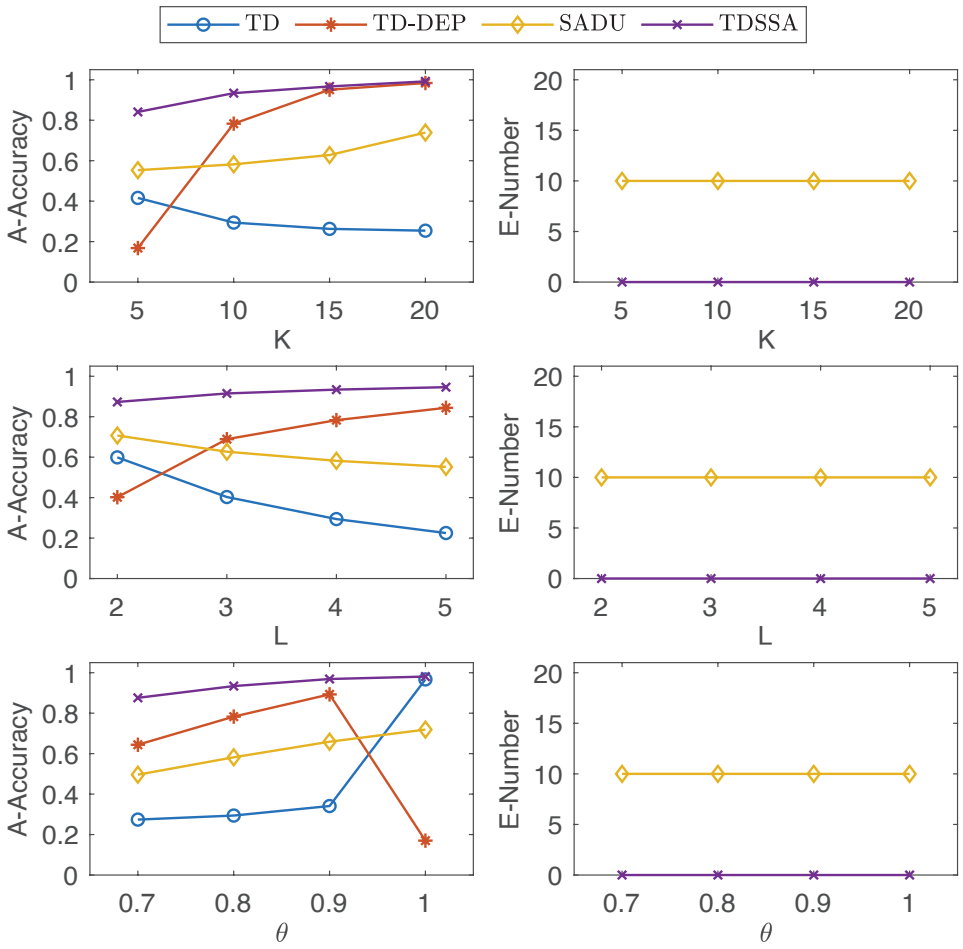

Figure 4.5: A-accuracy and E-number vs data parameters (K, L, θ) on the Synthetic datasets. We vary one parameter at a time with $K = 10$, $L = 4$ and $\theta = 0.8$ by default.

We also examined *TDSSA*'s failure rate of finding a valid golden task in Algorithm 5. Under the default settings, this rate is close to 0 for NLP and 0.92 for DOG, respectively. NLP contains more high-quality independent workers and there is more task promotion than DOG. Despite the high failure rating with DOG, *TDSSA*'s A-accuracy is still high because the Sybil scores and reliability scores obtained in early batches continue suppressing the impact of Sybil workers in subsequent batches after golden tasks were run out.

In summary, TDSSA is robust to Sybil attack with a consistently high A-accuracy and a small drop for up to 0.6 Sybil proportion. The low E-number and T-cost of TDSSA indicate that our probabilistic task assignment efficiently utilizes and well camouflages golden tasks.

4.7.4 Experiment on Synthetic Datasets

Figure 4.5 shows the A-accuracy and E-number vs varying data parameters (K, L, θ). Figure 4.6 reported the running time vs the task number N and worker number M.

A-Accuracy. *TDSSA* achieves the highest A-accuracy among all algorithms. When K increases, Sybil workers have a higher chance to form a false majority on golden tasks, thus, a higher chance to be caught through their Sybil scores, which improves *TDSSA*'s accuracy. A larger K allows more Sybil workers to work on a task and dominate the aggregation result, so the A-accuracy of *TD* decreases. Assigning more workers to a task provides more data for the similarity/dependency based *SADU* and *TD-DEP*, so their A-accuracy is improved.

When L increases, the A-accuracy of *TDSSA* is improved because Sybil workers have a higher probability $\frac{L-1}{L}$ to provide a false label on golden tasks, thus, easier to capture the Sybil behavior. For *TD* and *SADU*, A-accuracy decreases because Sybil workers have a higher chance to dominate the aggregation with a false label. For *TD-DEP*, a larger L reduces the chance for independent workers to provide the same false label on a task, which lowers their dependency. Therefore, the A-accuracy is improved.

In general, a larger worker accuracy θ will increase all algorithms' A-accuracy. At the maximum $\theta = 1$, independent workers agreed on the true label of each task and *TD* achieves almost the same A-accuracy as *TDSSA*. With the default $\mu = 0.5$, Sybil workers and independent workers have the same probability to dominate the aggregation result, but due to the $\epsilon = 0.1$ deviation probability, more tasks would be dominated by independent workers in the first iteration, which enables independent workers to gain more weight and exert more influence subsequently. At $\theta = 1$, *TD-DEP* fails to distinguish the label convergence of independent workers from the label sharing of Sybil workers, so its A-accuracy significantly drops.

E-Number. *SADU* has the E-number of 10 because most Sybil workers are clustered into the same group, exposing the 10 golden tasks used for evaluating the group quality. *TDSSA* has the zero E-number thanks to the camouflage of golden tasks through the restriction on the number of assignments for each golden task.

T-Cost. *SADU* has the fixed T-cost of 10 because each cluster is tested by 10 golden tasks. The T-cost of *TDSSA* is less than 1. In fact, 65% percent workers were not tested by any golden task in *TDSSA* after the initial 20 golden tasks were run out and no completed task could be promoted. In this case, the weight estimation of untested workers will still be affected by those tested workers through commonly labeled tasks in the extended truth discovery.

Runtime. *TDSSA* has a longer running time compared to *TD* because of executing multiple batches, but its complexity is bounded by $O(N^2)$ for the task number N and could become $O(N)$ if we apply a constant number of batches, e.g., terminating a batch based on the percentage of normal tasks in T being completed. *SADU* and *TD-DEP* have their running time linearly increased as N becomes larger, but due to the computation of the similarity or dependency between workers, both methods take much more running time than *TDSSA*. For the same reason, the running time of *SADU* and *TD-DEP* is very sensitive to the worker number M, while the complexity of *TDSSA* is independent of M.

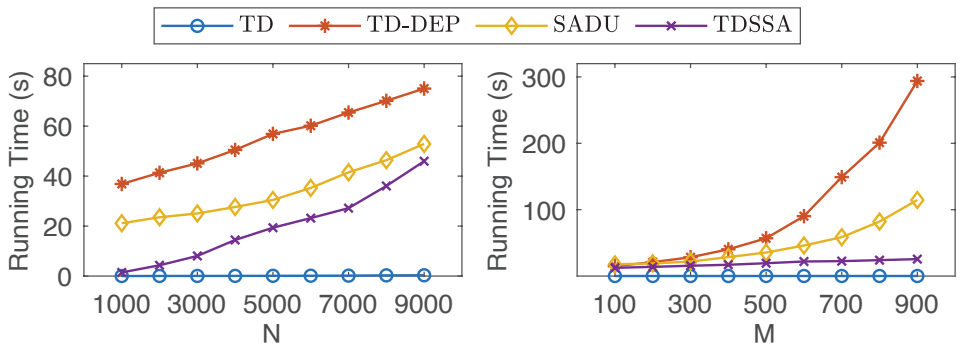

Figure 4.6: Running time vs task number N and worker number M on the Synthetic datasets. We vary one parameter at a time with $N = 5000$ and $M = 500$ by default.

Again, TDSSA maintains a consistently higher A-accuracy on synthetic datasets with varying characteristics. TDSSA is more efficient than the other methods. While TDSSA has a longer running time than TD, this pay is justified by the big improvement on A-accuracy.

4.8 Summary

In this chapter, we consider Sybil attack in crowdsourcing, where Sybil workers are coordinated to earn easy rewards by sharing randomized labels to outvote independent workers. With the assumption that a strategic Sybil attacker may attempt to evade detection by adaptively switching from a randomized label to an honest one when a golden task is identified and allowing Sybil workers to occasionally deviate from sharing a label, we formulate a more challenging problem of Sybil defense.

To solve the problem, we propose a *TDSSA* framework that gracefully incorporates Sybil behavior and labeling reliability of workers into truth discovery, instead of relying on a black-and-white detection of Sybil workers that suffers from the dilemma of having both high precision and high recall. We also addressed two related issues in this approach, i.e., camouflaging golden tasks from the attacker and the possible shortage of golden tasks.

Experiments on real datasets shows that *TDSSA* achieved consistently higher aggregation accuracy than state-of-the-art methods under different attack scenarios as we vary the percentage of Sybil workers in the system, the probability for Sybil workers to deviate from sharing labels and the number of strategic Sybil workers. We also create synthetic datasets with varying characteristics, such as task number, worker number, assignments per task, optional label size and average accuracy of online workers, to demonstrate the robustness and computational effectiveness of *TDSSA* against Sybil attack.

Chapter 5

Conclusion

5.1 Summary

Crowdsourcing has become a popular method for the crowd to collectively contribute their expertise to help solve human intelligent tasks (HITs). Despite the flexible workforce and tremendous profit potential, crowdsourcing faces many challenges due to its open nature. In this thesis, we proposed some new frameworks that help the HIT requester better handle the quality control and cost management in crowdsourcing, which are two major research directions for academic and industrial communities. The contributions of this thesis can be summarized as follows:

- In Chapter 3, we proposed an online framework GAA and an iterative version $IGAA$ to guarantee a specific accuracy for the aggregated labels while minimizing the worker cost. With the identification of candidate workers, our approach passes all the different thresholds of aggregation accuracy in the experiment and requires much less worker cost than the baseline methods. Importantly, our performance only depends on the quality of active workers, instead of their requesting order. To deal with the online setting, the parameters used for defining candidate workers should be updated w.r.t. the change of active workers, but our update has much less computational cost than other methods that also require periodic updates, e.g., iCrowd [26]. For simplicity, we only define candidate workers based on majority voting in this thesis and plan to apply the concept in other aggregation methods later.

- In Chapter 4, we proposed a Sybil defense framework $TDSSA$ that learns workers' Sybil behavior and labeling reliability from a well-camouflaged distribution of golden tasks. Experiments on real and synthetic datasets indicate that $TDSSA$ achieves zero exposure of golden tasks to the strategic Sybil attacker in most cases and provides much higher aggregation accuracy than baseline methods when the percentage of Sybil workers in the system becomes larger. Compared to other frameworks that analyze the label similarity between workers, e.g., $TD-DEP$ [21] and $SADU$ [102], $TDSSA$

requires lower computational cost to maintain stably high aggregation accuracy. A future improvement for $TDSSA$ is to consider worse attacking scenarios, where the attacker may have more knowledge about the system so that more strategies could be applied to evade detection.

5.2 Future Directions

Several future directions of deployment and research could be derived from this thesis. We list some interesting ones as follows.

5.2.1 Implementation of $GAA/IGAA$ and $TDSSA$ Frameworks

Most crowdsourcing platforms, e.g., Amazon Mechanical Turk (AMT)[1], do not allow the HIT requester to control over the online task assignment. Instead, when a worker requests, one or more tasks are randomly assigned to the worker, as long as the worker passes the qualification test. However, both of our $GAA/IGAA$ and $TDSSA$ frameworks require the selection of proper tasks assigned to each requesting worker to improve the aggregation accuracy. Fortunately, AMT also provides a feature called External Question [79], which displays an URL that links an requesting worker to the external website of the HIT requester. The web page will show a task as a form for the worker to fill out and submit. Then the form will send the answer back to AMT. In this way, the HIT requester will be able to decide which task to assign and how to collect worker answers.

To integrate the $GAA/IGAA$ and $TDSSA$ frameworks into our external web server, we will create a database to hold tasks, worker ID and worker answers. When a worker links to our website from AMT, we check the ID of the worker and evaluate his historical performance on golden tasks to estimate worker quality. Then, we run the proposed algorithms to decide which task the requesting worker will be assigned to and display it to the worker. Different aggregation methods could also be applied on the server to infer the true answer of tasks.

In addition to the implementation of $GAA/IGAA$ and $TDSSA$ frameworks, we can add other features on our web server to improve the performance. First, a better user interface could be designed to increase the efficiency for workers to answer tasks. Second, an IP check would be provided to detect workers from the same IP address, which increases the cost for a Sybil attacker to manipulate several robot Sybil workers. Last, we should also look into security issues as the attacker might try to hack our web server to copy answers given by independent workers, instead of providing randomized but probably wrong answers for Sybil workers to share.

[1] https://www.mturk.com/

5.2.2 Accuracy Guarantee and Cost Minimization under Group Attack

In Chapter 3, we mainly focus on the quality control and cost management under individual attack, where workers provide their labels independently. One interesting future work is whether we could also perform task assignment based on worker quality to guarantee a specific aggregation accuracy with minimum worker cost under group attack, such as Sybil attack and data poisoning attack.

Providing an accuracy guarantee with minimized worker cost under Sybil attack is challenging. Since the worker accuracy estimated by golden tasks does not reflect the real expertise of Sybil workers, we cannot simply adopt the evaluation of inference probability and the definition of candidate workers in our $GAA/IGAA$ frameworks. Second, both $GAA/IGAA$ and $TDSSA$ we proposed in Chapter 3 and 4 use golden tasks to reveal workers' quality and/or Sybil behavior due to the absence of a priori information about dynamic online workers, but the golden task assignment is applied in different ways. Unlike $GAA/IGAA$ that apply a cold-start phase to estimate each new worker's task-specific accuracy using golden tasks when the worker arrives, $TDSSA$ performs the distribution and camouflage of golden tasks as a hidden test to gradually reveal worker quality.

For data poisoning attack, guaranteeing the aggregation accuracy with the minimum worker cost is even harder to achieve as the attacker may have full or partial knowledge about the system, e.g., the labels provided by normal workers and the algorithm for aggregating worker labels, to maximize the errors in the aggregation result. Similar to our $TDSSA$ framework, a possible solution is to evaluate the influence of each worker on the aggregation result and apply golden tasks to test workers with the maximum influence. In this case, the chance for the exposure of golden tasks to the attacker would be reduced.

The use of golden tasks for estimating worker quality does not only incur more cost but also introduces the risk of being manipulated by the group attacker to purposely improve the reputation of malicious workers. Recently, Multi-armed bandit methods [72] have been recommended for eliminating golden tasks. The idea is to apply a reinforcement learning model that exemplifies the exploration–exploitation trade-off dilemma, where new information about online workers is acquired (exploration) and the task assignment based on existing information is optimized (exploitation). Multi-armed bandit algorithms provide a potential solution for integrating the two problems we studied in this thesis.

5.2.3 Neural Network for Sybil Defense in Crowdsourcing

We mentioned in Chapter 4 that most works conduct a similarity-based approach for copy detection [21, 55, 106] or Sybil detection [102]. According to the labels provided by workers on their common tasks, the similarity between each pair of workers is measured. Then, different methods are applied to reduce the contribution of similar workers with low quality. However, if a strategic Sybil attacker purposely limits the number of common tasks labeled

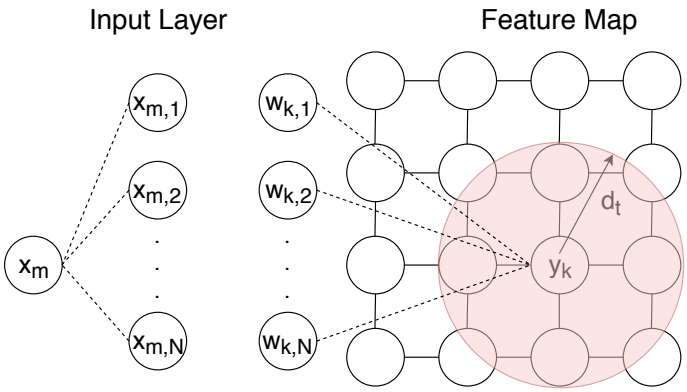

Figure 5.1: An example of Self Organizing Map

by each pair of Sybil workers, their explicit similarity would be weaken. One interesting future work is whether we could also reveal the latent correlation between online workers for Sybil defense when such a strategy of controlling common tasks is applied.

As the goal of Sybil attack is to outvote independent workers through the sharing of randomized labels among several Sybil workers, it is impossible for the labels provided by Sybil workers to deviate from correlation, which could be well revealed by neural network. Neural Network has shown its great power in image recognition, speech recognition, and natural language processing. Recently, a heterogeneous graph neural network model [99] was proposed to take into account the latent worker correlation and task correlation. A drawback of this model is that the learning procedure is conducted by building a worker-task assignment graph after collecting worker labels, so we cannot identify or ban Sybil workers as early as possible to save budget for collecting labels from independent workers.

To utilize the latent worker correlation for improving the online task assignment, an unsupervised neural network learning algorithm called Self-Organizing Map (SOM) [50] may be applied to capture the correlation of Sybil workers during online task assignment. An SOM model is a feature map with a set of interconnected neurons, as shown in Figure 5.1. Let y_k denote a neuron on the map, N denote the task size, $\{w_{k,1}, ..., w_{k,N}\}$ denote a weight vector associated to neuron y_k, and $x_m = \{x_{m,1}, ..., x_{m,N}\}$ denote the labels provided by a worker m. Each time a worker submits a label, the input vector x_m is updated and presented to the model. By comparing with the weight vector of each neuron, we can locate the Best Matching Unit (BMU) having its weight vector closest to x_m. Then, the weight vectors of the BMU and its neighboring neurons are updated towards the input vector.

The SOM model aims to achieve a topological mapping between the input data and the neurons. By gradually mapping Sybil workers into the same or topologically close neurons on the map, the weight vector of these neurons will become close to the attacker's randomly generated labels from which Sybil workers sample their labels on tasks. It allows us to reveal the correlation between Sybil workers during online task assignment, without explicitly

evaluating their pairwise similarity. In addition, the SOM model can handle the presence of multiple Sybil attackers because Sybil workers under different attackers' control would be mapped into neurons with diverse weight vectors, which are topologically grouped in different areas on the feature map.

Bibliography

[1] Ittai Abraham, Omar Alonso, Vasilis Kandylas, Rajesh Patel, Steven Shelford, and Aleksandrs Slivkins. How many workers to ask?: Adaptive exploration for collecting high quality labels. In *SIGIR*, pages 473–482. ACM, 2016.

[2] Eman Aldhahri, Vivek Shandilya, and Sajjan Shiva. Towards an effective crowdsourcing recommendation system: A survey of the state-of-the-art. In *2015 IEEE Symposium on Service-Oriented System Engineering*, pages 372–377. IEEE, 2015.

[3] Florian Alt, Alireza Sahami Shirazi, Albrecht Schmidt, Urs Kramer, and Zahid Nawaz. Location-based crowdsourcing: extending crowdsourcing to the real world. In *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*, pages 13–22, 2010.

[4] Vamshi Ambati, Stephan Vogel, and Jaime G Carbonell. Towards task recommendation in micro-task markets. *Human computation*, 11:1–4, 2011.

[5] Yael Amsterdamer, Susan B Davidson, Tova Milo, Slava Novgorodov, and Amit Somech. Oassis: query driven crowd mining. In *Proceedings of SIGMOD*, pages 589–600. ACM, 2014.

[6] Bahadir Ismail Aydin, Yavuz Selim Yilmaz, Yaliang Li, Qi Li, Jing Gao, and Murat Demirbas. Crowdsourcing for multiple-choice question answering. In *AAAI*, pages 2946–2953, 2014.

[7] Liliya I Besaleva and Alfred C Weaver. Crowdhelp: A crowdsourcing application for improving disaster management. In *2013 IEEE Global Humanitarian Technology Conference (GHTC)*, pages 185–190. IEEE, 2013.

[8] Wei Bi and James T Kwok. Mandatory leaf node prediction in hierarchical multilabel classification. *IEEE transactions on neural networks and learning systems*, 25(12):2275–2287, 2014.

[9] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Proceedings of JMLR*, 3:993 1022, 2003.

[10] Rubi Boim, Ohad Greenshpan, Tova Milo, Slava Novgorodov, Neoklis Polyzotis, and Wang-Chiew Tan. Asking the right questions in crowd data sourcing. In *2012 IEEE 28th International Conference on Data Engineering*, pages 1261–1264. IEEE, 2012.

[11] Ria Mae Borromeo and Motomichi Toyama. Automatic vs. crowdsourced sentiment analysis. In *Proceedings of the 19th International Database Engineering & Applications Symposium*, pages 90–95, 2015.

[12] Jonathan Bragg, Daniel S Weld, et al. Crowdsourcing multi-label classification for taxonomy creation. In *First AAAI conference on human computation and crowdsourcing*, 2013.

[13] Chris Callison-Burch. Fast, cheap, and creative: evaluating translation quality using amazon's mechanical turk. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 286–295. Association for Computational Linguistics, 2009.

[14] Sean X Chen and Jun S Liu. Statistical applications of the poissonbinomial and conditional bernoulli distributions. *Statistica Sinica*, 7(4):875 892, 1997.

[15] Xi Chen, Qihang Lin, and Dengyong Zhou. Optimistic knowledge gradient policy for optimal budget allocation in crowdsourcing. In *Proceedings of ICML*, pages 64–72, 2013.

[16] XiangHui Chen, Arthur P Dempster, and Jun S Liu. Weighted finite population sampling to maximize entropy. *Biometrika*, 81(3):457 469, 1994.

[17] Dan Cosley, Shyong K Lam, Istvan Albert, Joseph A Konstan, and John Riedl. Is seeing believing? how recommender system interfaces affect users' opinions. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 585–592, 2003.

[18] Alexander Philip Dawid and Allan M Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied statistics*, pages 20–28, 1979.

[19] Gianluca Demartini, Djellel Eddine Difallah, and Philippe Cudré-Mauroux. Zencrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In *World Wide Web*, pages 469–478. ACM, 2012.

[20] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[21] Xin Luna Dong, Laure Berti-Equille, and Divesh Srivastava. Integrating conflicting data: the role of source dependence. *VLDB*, 2(1):550–561, 2009.

[22] Xin Luna Dong, Laure Berti-Equille, and Divesh Srivastava. Truth discovery and copying detection in a dynamic world. *VLDB*, 2(1):562–573, 2009.

[23] John R Douceur. The sybil attack. In *International workshop on peer-to-peer systems*, pages 251–260. Springer, 2002.

[24] Xiaoni Duan and Keishi Tajima. Improving multiclass classification in crowdsourcing by using hierarchical schemes. In *WWW*, pages 2694–2700, 2019.

[25] Cynthia Dwork. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pages 1–19. Springer, 2008.

[26] Ju Fan, Guoliang Li, Beng Chin Ooi, Kian-lee Tan, and Jianhua Feng. icrowd: An adaptive crowdsourcing framework. In *SIGMOD*, pages 1015–1030. ACM, 2015.

[27] Minghong Fang, Minghao Sun, Qi Li, Neil Zhenqiang Gong, Jin Tian, and Jia Liu. Data poisoning attacks and defenses to crowdsourcing systems. *arXiv preprint arXiv:2102.09171*, 2021.

[28] Rita Faullant, Johann Fueller, and Katja Hutter. Fair play: perceived fairness in crowdsourcing communities and its behavioral consequences. In *Academy of Management Proceedings*, volume 2013, page 15433. Academy of Management Briarcliff Manor, NY 10510, 2013.

[29] Nikolaus Franke, Peter Keinz, and Katharina Klausberger. "does this sound like a fair deal?": Antecedents and consequences of fairness expectations in the individual's decision to participate in firm innovation. *Organization science*, 24(5):1495–1516, 2013.

[30] Michael J Franklin, Donald Kossmann, Tim Kraska, Sukriti Ramesh, and Reynold Xin. Crowddb: answering queries with crowdsourcing. In *SIGMOD*, pages 61–72, 2011.

[31] Alban Galland, Serge Abiteboul, Amélie Marian, and Pierre Senellart. Corroborating information from disagreeing views. In *WSDM*, pages 131–140. ACM, 2010.

[32] Yanmin Gong, Lingbo Wei, Yuanxiong Guo, Chi Zhang, and Yuguang Fang. Optimal task recommendation for mobile crowdsourcing with privacy control. *IEEE Internet of Things Journal*, 3(5):745–756, 2015.

[33] Thomas L Griffiths and Mark Steyvers. Finding scientific topics. *Proceedings of the National academy of Sciences*, 101(suppl 1):5228–5235, 2004.

[34] Tao Han, Hailong Sun, Yangqiu Song, Zizhe Wang, and Xudong Liu. Budgeted task scheduling for crowdsourced knowledge acquisition. In *Proceedings of CIKM*, pages 1059–1068. ACM, 2017.

[35] Ann Hillier, Ryan P Kelly, and Terrie Klinger. Narrative style influences citation frequency in climate change science. *PloS one*, 11(12), 2016.

[36] Chien-Ju Ho, Shahin Jabbari, and Jennifer Wortman Vaughan. Adaptive task assignment for crowdsourced classification. In *ICML*, pages 534–542, 2013.

[37] Jeff Howe. The rise of crowdsourcing. *Wired magazine*, 14(6):1–4, 2006.

[38] Zehong Hu and Jie Zhang. Optimal posted-price mechanism in microtask crowdsourcing. In *IJCAI*, pages 228–234, 2017.

[39] Srikanth Jagabathula, Lakshminarayanan Subramanian, and Ashwin Venkataraman. Reputation-based worker filtering in crowdsourcing. In *Advances in Neural Information Processing Systems*, pages 2492–2500, 2014.

[40] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 19–35. IEEE, 2018.

[41] Lingyun Jiang, Xiaofu Niu, Jia Xu, Dejun Yang, and Lijie Xu. Incentivizing the workers for truth discovery in crowdsourcing with copiers. *arXiv preprint arXiv:1902.03889*, 2019.

[42] Xiaocong Jin and Yanchao Zhang. Privacy-preserving crowdsourced spectrum sensing. *IEEE/ACM Transactions on Networking*, 26(3):1236–1249, 2018.

[43] Parisa Kaghazgaran, James Caverlee, and Anna Squicciarini. Combating crowdsourced review manipulators: a neighborhood-based approach. In *WSDM*, pages 306–314. ACM, 2018.

[44] Haim Kaplan, Ilia Lotosh, Tova Milo, and Slava Novgorodov. Answering planning queries with the crowd. *Proceedings of VLDB*, 6(9):697–708, 2013.

[45] David R Karger, Sewoong Oh, and Devavrat Shah. Iterative learning for reliable crowdsourcing systems. In *Advances in neural information processing systems*, pages 1953–1961, 2011.

[46] Leyla Kazemi and Cyrus Shahabi. A privacy-aware framework for participatory sensing. *ACM Sigkdd Explorations Newsletter*, 13(1):43–51, 2011.

[47] Leyla Kazemi and Cyrus Shahabi. Towards preserving privacy in participatory sensing. In *2011 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 328–331. IEEE, 2011.

[48] Hyun-Chul Kim and Zoubin Ghahramani. Bayesian classifier combination. In *Artificial Intelligence and Statistics*, pages 619–627, 2012.

[49] Ari Kobren, Chun How Tan, Panagiotis Ipeirotis, and Evgeniy Gabrilovich. Getting more for less: Optimized crowdsourcing with dynamic tasks and goals. In *Proceedings of the 24th international conference on world wide web*, pages 592–602, 2015.

[50] Teuvo Kohonen. The self-organizing map. *Neurocomputing*, 21(1-3):1–6, 1998.

[51] Matthew Lease and Emine Yilmaz. Crowdsourcing for information retrieval. In *ACM SIGIR Forum*, volume 45, pages 66–75. ACM New York, NY, USA, 2012.

[52] Bo Li, Yining Wang, Aarti Singh, and Yevgeniy Vorobeychik. Data poisoning attacks on factorization-based collaborative filtering. *arXiv preprint arXiv:1608.08182*, 2016.

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

[54] Qi Li, Yaliang Li, Jing Gao, Bo Zhao, Wei Fan, and Jiawei Han. Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation. In *SIGMOD*, pages 1187–1198. ACM, 2014.

[55] Xian Li, Xin Luna Dong, Kenneth B Lyons, Weiyi Meng, and Divesh Srivastava. Scaling up copy detection. In *ICDE*, pages 89–100. IEEE, 2015.

[56] Yaliang Li, Jing Gao, Chuishi Meng, Qi Li, Lu Su, Bo Zhao, Wei Fan, and Jiawei Han. A survey on truth discovery. *ACM Sigkdd Explorations Newsletter*, 17(2):1–16, 2016.

[57] Yaliang Li, Chenglin Miao, Lu Su, Jing Gao, Qi Li, Bolin Ding, Zhan Qin, and Kui Ren. An efficient two-layer mechanism for privacy-preserving truth discovery. In *SIGKDD*, pages 1705–1714. ACM, 2018.

[58] An Liu, Weiqi Wang, Shuo Shang, Qing Li, and Xiangliang Zhang. Efficient task assignment in spatial crowdsourcing with worker and task privacy protection. *GeoInformatica*, 22(2):335–362, 2018.

[59] Bozhong Liu, Ling Chen, Xingquan Zhu, Ying Zhang, Chengqi Zhang, and Weidong Qiu. Protecting location privacy in spatial crowdsourcing using encrypted data. *Advances in Database Technology-EDBT*, 2017.

[60] Qiang Liu, Jian Peng, and Alexander T Ihler. Variational inference for crowdsourcing. In *Advances in neural information processing systems*, pages 692–700, 2012.

[61] Xuan Liu, Meiyu Lu, Beng Chin Ooi, Yanyan Shen, Sai Wu, and Meihui Zhang. Cdas: a crowdsourcing data analytics system. *VLDB*, 5(10):1040–1051, 2012.

[62] Fenglong Ma, Yaliang Li, Qi Li, Minghui Qiu, Jing Gao, Shi Zhi, Lu Su, Bo Zhao, Heng Ji, and Jiawei Han. Faitcrowd: Fine grained truth discovery for crowdsourced data aggregation. In *SIGKDD*, pages 745–754. ACM, 2015.

[63] Adam Marcus, David Karger, Samuel Madden, Robert Miller, and Sewoong Oh. Counting with the crowd. In *VLDB*, volume 6, pages 109–120. VLDB Endowment, 2012.

[64] Adam Marcus, Eugene Wu, David R Karger, Samuel Madden, and Robert C Miller. Crowdsourced databases: query processing with people. In *CIDR*. CIDR, 2011.

[65] Chenglin Miao, Qi Li, Lu Su, Mengdi Huai, Wenjun Jiang, and Jing Gao. Attack under disguise: an intelligent data poisoning attack mechanism in crowdsourcing. In *WWW*, pages 13–22. IW3C2, 2018.

[66] Chenglin Miao, Qi Li, Houping Xiao, Wenjun Jiang, Mengdi Huai, and Lu Su. Towards data poisoning attacks in crowd sensing systems. In *Proceedings of the Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 111–120. ACM, 2018.

[67] Stefanie Nowak and Stefan Rüger. How reliable are annotations via crowdsourcing: a study about inter-annotator agreement for multi-label image annotation. In *Proceedings of the international conference on Multimedia information retrieval*, pages 557–566, 2010.

[68] David Oleson, Alexander Sorokin, Greg Laughlin, Vaughn Hester, John Le, and Lukas Biewald. Programmatic gold: Targeted and scalable quality assurance in crowdsourcing. In *Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[69] Aditya G Parameswaran, Hector Garcia-Molina, Hyunjung Park, Neoklis Polyzotis, Aditya Ramesh, and Jennifer Widom. Crowdscreen: algorithms for filtering data with humans. In *SIGMOD*, pages 361–372. ACM, 2012.

[70] Layla Pournajaf, Li Xiong, Vaidy Sunderam, and Slawomir Goryczka. Spatial task assignment for crowd sensing with cloaked locations. In *2014 IEEE 15th International Conference on Mobile Data Management*, volume 1, pages 73–82. IEEE, 2014.

[71] Chenxi Qiu, Anna Squicciarini, and Benjamin Hanrahan. Incentivizing distributive fairness for crowdsourcing workers. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 404–412, 2019.

[72] Anshuka Rangi and Massimo Franceschetti. Multi-armed bandit algorithms for crowdsourcing systems with online estimation of workers' ability. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1345–1352, 2018.

[73] Vikas C Raykar and Shipeng Yu. Eliminating spammers and ranking annotators for crowdsourced labeling tasks. *Journal of Machine Learning Research*, 13(Feb):491–518, 2012.

[74] Vikas C Raykar, Shipeng Yu, Linda H Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. Learning from crowds. *Journal of Machine Learning Research*, 11(Apr):1297–1322, 2010.

[75] Xuebin Ren, Chia-Mu Yu, Weiren Yu, Shusen Yang, Xinyu Yang, Julie A McCann, and S Yu Philip. Lopub: High-dimensional crowdsourced data publication with local differential privacy. *IEEE Transactions on Information Forensics and Security*, 13(9):2151–2166, 2018.

[76] Senjuti Basu Roy, Ioanna Lykourentzou, Saravanan Thirumuruganathan, Sihem Amer-Yahia, and Gautam Das. Task assignment optimization in knowledge-intensive crowdsourcing. *The VLDB Journal*, 24(4):467–491, 2015.

[77] Marta Sabou, Kalina Bontcheva, and Arno Scharl. Crowdsourcing research opportunities: lessons from natural language processing. In *Proceedings of the 12th International Conference on Knowledge Management and Knowledge Technologies*, pages 1–8, 2012.

[78] Steffen Schnitzer, Christoph Rensing, Sebastian Schmidt, Kathrin Borchert, Matthias Hirth, and Phuoc Tran-Gia. Demands on task recommendation in crowdsourcing platforms-the worker's perspective. In *CrowdRec Workshop*, 2015.

[79] Amazon Web Services. Amazon mechanical turk (external question). `https://docs.aws.amazon.com/AWSMechTurk/latest/AWSMturkAPI/ApiReference_ExternalQuestionArticle.html`. Accessed: 2021-02-07.

[80] Padhraic Smyth, Usama Fayyad, Michael Burl, Pietro Perona, and Pierre Baldi. Inferring ground truth from subjective labelling of venus images. *Advances in neural information processing systems*, 7:1085–1092, 1994.

[81] Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y Ng. Cheap and fast - but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pages 254–263. ACL, 2008.

[82] Hien To, Gabriel Ghinita, and Cyrus Shahabi. A framework for protecting worker location privacy in spatial crowdsourcing. *Proceedings of the VLDB Endowment*, 7(10):919–930, 2014.

[83] Evangelos Tripolitakis and Georgios Chalkiadakis. Probabilistic topic modeling, reinforcement learning, and crowdsourcing for personalized recommendations. In *Multi-Agent Systems and Agreement Technologies*, pages 157–171. Springer, 2016.

[84] Jennifer Wortman Vaughan. Making better use of the crowd: How crowdsourcing can advance machine learning research. *J. Mach. Learn. Res.*, 18(1):7026–7071, 2017.

[85] Matteo Venanzi, John Guiver, Gabriella Kazai, Pushmeet Kohli, and Milad Shokouhi. Community-based bayesian aggregation models for crowdsourcing. In *WWW*, pages 155–164. ACM, 2014.

[86] Norases Vesdapunt, Kedar Bellare, and Nilesh Dalvi. Crowdsourcing algorithms for entity resolution. *Proceedings of VLDB*, 7(12):1071–1082, 2014.

[87] Luis Von Ahn, Benjamin Maurer, Colin McMillen, David Abraham, and Manuel Blum. recaptcha: Human-based character recognition via web security measures. *Science*, 321(5895):1465–1468, 2008.

[88] Khuong Vu, Rong Zheng, and Jie Gao. Efficient algorithms for k-anonymous location privacy in participatory sensing. In *2012 Proceedings IEEE INFOCOM*, pages 2399–2407. IEEE, 2012.

[89] Jeroen Vuurens, Arjen P de Vries, and Carsten Eickhoff. How much spam can you take? an analysis of crowdsourcing results to increase accuracy. In *Proc. ACM SIGIR Workshop on Crowdsourcing for Information Retrieval (CIR'11)*, pages 21–26, 2011.

[90] Gang Wang, Bolun Wang, Tianyi Wang, Ana Nika, Haitao Zheng, and Ben Y Zhao. Defending against sybil devices in crowdsourced mapping services. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, pages 179–191. ACM, 2016.

[91] Gang Wang, Bolun Wang, Tianyi Wang, Ana Nika, Haitao Zheng, and Ben Y Zhao. Ghost riders: Sybil attacks on crowdsourced mobile mapping services. *IEEE/ACM transactions on networking*, 2018.

[92] Jiannan Wang, Tim Kraska, Michael J Franklin, and Jianhua Feng. Crowder: Crowdsourcing entity resolution. *Proceedings of VLDB*, 5(11):1483–1494, 2012.

[93] Jiannan Wang, Guoliang Li, Tim Kraska, Michael J Franklin, and Jianhua Feng. Leveraging transitive relations for crowdsourced joins. In *Proceedings of SIGMOD*, pages 229–240. ACM, 2013.

[94] Leye Wang, Daqing Zhang, Dingqi Yang, Brian Y Lim, and Xiaojuan Ma. Differential location privacy for sparse mobile crowdsensing. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 1257–1262. IEEE, 2016.

[95] Sibo Wang, Xiaokui Xiao, and Chun-Hee Lee. Crowd-based deduplication: An adaptive approach. In *Proceedings of SIGMOD*, pages 1263–1277. ACM, 2015.

[96] Peter Welinder, Steve Branson, Pietro Perona, and Serge J Belongie. The multidimensional wisdom of crowds. In *Advances in neural information processing systems*, pages 2424–2432, 2010.

[97] Steven Euijong Whang, Peter Lofgren, and Hector Garcia-Molina. Question selection for crowd entity resolution. *Proceedings of VLDB*, 6(6):349–360, 2013.

[98] Jacob Whitehill, Ting-fan Wu, Jacob Bergsma, Javier R Movellan, and Paul L Ruvolo. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Advances in neural information processing systems*, pages 2035–2043, 2009.

[99] Hanlu Wu, Tengfei Ma, Lingfei Wu, and Shouling Ji. Exploiting heterogeneous graph neural networks with latent worker/task correlation information for label aggregation in crowdsourcing. *arXiv preprint arXiv:2010.13080*, 2020.

[100] Jingru Yang, Ju Fan, Zhewei Wei, Guoliang Li, Tongyu Liu, and Xiaoyong Du. Cost-effective data annotation using game-based crowdsourcing. *Proceedings of VLDB*, 12(4), 2018.

[101] Ming Yin and Yiling Chen. Bonus or not? learn to reward in crowdsourcing. In *IJCAI*, pages 201–208, 2015.

[102] Dong Yuan, Guoliang Li, Qi Li, and Yudian Zheng. Sybil defense in crowdsourcing platforms. In *CIKM*, pages 1529–1538. ACM, 2017.

[103] Man-Ching Yuen, Irwin King, and Kwong-Sak Leung. Task matching in crowdsourcing. In *2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing*, pages 409–412. IEEE, 2011.

[104] Omar F Zaidan and Chris Callison-Burch. Crowdsourcing translation: Professional quality from non-professionals. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1220–1229. Association for Computational Linguistics, 2011.

[105] Dongjun Zhai, Yue Sun, An Liu, Zhixu Li, Guanfeng Liu, Lei Zhao, and Kai Zheng. Towards secure and truthful task assignment in spatial crowdsourcing. *World Wide Web*, 22(5):2017–2040, 2019.

[106] Hengtong Zhang, Qi Li, Fenglong Ma, Houping Xiao, Yaliang Li, Jing Gao, and Lu Su. Influence-aware truth discovery. In *CIKM*, pages 851–860. ACM, 2016.

[107] Xinglin Zhang, Longfei Shangguan, and Ye Yuan. A crowd wisdom management framework for crowdsourcing systems. *IEEE Access*, 4:9764–9774, 2016.

[108] Haizhong Zheng, Minhui Xue, Hao Lu, Shuang Hao, Haojin Zhu, Xiaohui Liang, and Keith Ross. Smoke screener or straight shooter: detecting elite sybil attacks in user-review social networks. *arXiv preprint arXiv:1709.06916*, 2017.

[109] Yudian Zheng, Guoliang Li, and Reynold Cheng. Docs: a domain-aware crowdsourcing system using knowledge bases. *VLDB*, 10(4):361–372, 2016.

[110] Yudian Zheng, Guoliang Li, Yuanbing Li, Caihua Shan, and Reynold Cheng. Truth inference in crowdsourcing: Is the problem solved? *VLDB*, 10(5):541–552, 2017.

[111] Yudian Zheng, Jiannan Wang, Guoliang Li, Reynold Cheng, and Jianhua Feng. Qasca: A quality-aware task assignment system for crowdsourcing applications. In *SIGMOD*, pages 1031–1046. ACM, 2015.

[112] Dengyong Zhou, Sumit Basu, Yi Mao, and John C Platt. Learning from the wisdom of crowds by minimax entropy. In *Advances in neural information processing systems*, pages 2195–2203, 2012.