8-2018

Spam elimination and bias correction : ensuring label quality in crowdsourced tasks.

Lingyu Lyu
University of Louisville

SPAM ELIMINATION AND BIAS CORRECTION: ENSURING LABEL QUALITY IN CROWDSOURCED TASKS

By

Lingyu Lyu
M.S., University of Louisville, USA, 2014
B.Tech., Soochow University, China, 2011

A Dissertation
Submitted to the Faculty of the
J. B. Speed School of Engineering at the University of Louisville
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy
in Computer Science and Engineering

Department of Computer Engineering and Computer Science
University of Louisville
Louisville, Kentucky

August 2018

SPAM ELIMINATION AND BIAS CORRECTION: ENSURING LABEL QUALITY IN CROWDSOURCED TASKS

By

Lingyu Lyu
M.S., University of Louisville, USA, 2014
B.Tech., Soochow University, China, 2011

A Dissertation Approved On

June 28, 2018

by the following Dissertation Committee:

Mehmed Kantardzic, Ph.D., Dissertation Director

Adel Elmaghraby, Ph.D.

Adrian Lauf, Ph.D.

Ibrahim Imam, Ph.D.

James Lewis, Ph.D.

DEDICATION

To my parents, for their unconditional love, and to my sister, for her supports.

ACKNOWLEDGEMENTS

The path toward completion of this dissertation has been circuitous and challenging. It would not have been possible without the support and guidance that I received from many people.

I would like to express my special appreciation and thanks to my advisor Dr. Mehmed Kantardzic for his valuable guidance, support, and consistent encouragement I received throughout the doctoral studies. He has provided great inspiration and insightful suggestions about my research. His expertise, understanding, and patience helped me throughout all of my work. Without his persistent help, this PhD would not have been achievable.

Many thanks also to my committee members - Dr. Adel Elmaghraby, Dr. Adrian Lauf, Dr. Ibrahim Imam, and Dr. James Lewis, for their great support and constructive advice. Their helpful comments and professional feedback have helped me develop a broader perspective to my dissertation.

I would also like to thank all the members in Data Mining Lab for generously sharing their time and knowledge in our cooperative work. The professors and staffs in the University of Louisville and Department of Computer Engineering and Computer Science, have been helpful and friendly, which made the years in school remarkable.

I am grateful to my friends for helping me get through the difficult times, and for all the emotional support, camaraderie, and caring they provided. I would especially like to thank Lucas Trost, for cheering me on, and believing in me during various stages of my graduate studies.

My deepest gratitude to my family for their unconditional love, patience, and understanding in all my pursuits. My parents, Zhonghua Lyu and Zhumei Chen, and my sister Lingqi Lyu, have always been tolerant and supportive. I know I always have my family to count on when times are rough, and I would not have made it this far without them.

ABSTRACT

SPAM ELIMINATION AND BIAS CORRECTION: ENSURING LABEL QUALITY IN CROWDSOURCED TASKS

Lingyu Lyu

June 28, 2018

Crowdsourcing is proposed as a powerful mechanism for accomplishing large scale tasks via anonymous workers online. It has been demonstrated as an effective and important approach for collecting labeled data in application domains which require human intelligence, such as image labeling, video annotation, natural language processing, etc. Despite the promises, one big challenge still exists in crowdsourcing systems: the difficulty of controlling the quality of crowds. The workers usually have diverse education levels, personal preferences, and motivations, leading to unknown work performance while completing a crowdsourced task. Among them, some are reliable, and some might provide noisy feedback. It is intrinsic to apply worker filtering approach to crowdsourcing applications, which recognizes and tackles noisy workers, in order to obtain high-quality labels. The presented work in this dissertation provides discussions in this area of research, and proposes efficient probabilistic based worker filtering models to distinguish varied types of poor quality workers.

Most of the existing work in literature in the field of worker filtering either only concentrates on binary labeling tasks, or fails to separate the low quality workers whose label errors can be corrected from the other spam workers (with label errors which cannot be corrected). As such, we first propose a Spam Removing and De-biasing Framework (SRDF), to deal with the worker filtering procedure in labeling tasks with numerical label scales. The developed framework can detect spam workers and biased workers separately. The biased workers are defined as those who show tendencies of providing higher (or lower) labels than truths, and their errors are able to be corrected. To tackle the biasing problem, an iterative bias detection approach is introduced to recognize the biased workers. The spam filtering algorithm proposes to eliminate three types of spam workers, including

random spammers who provide random labels, uniform spammers who give same labels for most of the items, and sloppy workers who offer low accuracy labels. Integrating the spam filtering and bias detection approaches into aggregating algorithms, which infer truths from labels obtained from crowds, can lead to high quality consensus results.

The common characteristic of random spammers and uniform spammers is that they provide useless feedback without making efforts for a labeling task. Thus, it is not necessary to distinguish them separately. In addition, the removal of sloppy workers has great impact on the detection of biased workers, with the SRDF framework. To combat these problems, a different way of worker classification is presented in this dissertation. In particular, the biased workers are classified as a subcategory of sloppy workers. Finally, an ITerative Self Correcting – Truth Discovery (ITSC-TD) framework is then proposed, which can reliably recognize biased workers in ordinal labeling tasks, based on a probabilistic based bias detection model. ITSC-TD estimates true labels through applying an optimization based truth discovery method, which minimizes overall label errors by assigning different weights to workers.

The typical tasks posted on popular crowdsourcing platforms, such as MTurk, are simple tasks, which are low in complexity, independent, and require little time to complete. Complex tasks, however, in many cases require the crowd workers to possess specialized skills in task domains. As a result, this type of task is more inclined to have the problem of poor quality of feedback from crowds, compared to simple tasks. As such, we propose a multiple views approach, for the purpose of obtaining high quality consensus labels in complex labeling tasks. In this approach, each view is defined as a labeling critique or rubric, which aims to guide the workers to become aware of the desirable work characteristics or goals. Combining the view labels results in the overall estimated labels for each item. The multiple views approach is developed under the hypothesis that workers' performance might differ from one view to another. Varied weights are then assigned to different views for each worker. Additionally, the ITSC-TD framework is integrated into the multiple views model to achieve high quality estimated truths for each view.

Next, we propose a Semi-supervised Worker Filtering (SWF) model to eliminate spam workers, who assign random labels for each item. The SWF approach conducts worker filtering with a limited set of gold truths available as priori. Each worker is associated with a spammer score, which is estimated via the developed semi-supervised model, and low quality workers are efficiently detected by comparing the spammer score with a predefined threshold value.

The efficiency of all the developed frameworks and models are demonstrated on simulated and

real-world data sets. By comparing the proposed frameworks to a set of state-of-art methodologies, such as expectation maximization based aggregating algorithm, GLAD and optimization based truth discovery approach, in the domain of crowdsourcing, up to 28.0% improvement can be obtained for the accuracy of true label estimation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

CHAPTER 1

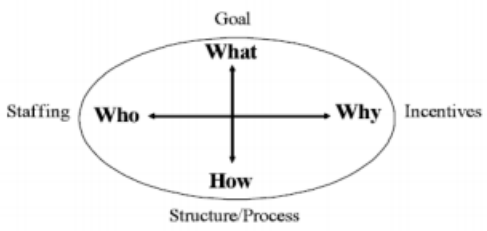
INTRODUCTION

1.1 Crowdsourcing

Crowdsourcing is defined as the practice of obtaining needed service, ideas, or contents by soliciting contributions from a large group of people, especially from the online community rather than from traditional employees or suppliers [20]. It is motivated by a fact that: a sufficiently large crowd of independent individuals will decide properly, with high probability, even if the individuals only have a slight bias towards the correct answer [9]. Crowdsourcing has been used in many areas such as crowdfunding, creative crowdsourcing, and crowd solving. For example, a company can use a crowdsourcing approach to request a logo design by posting the task on a crowdsourcing platform such as Amazon Mechanical Turk (MTurk). Gathering products reviews to provide information for producers is another way to make use of crowdsourcing. The guiding principle behind crowdsourcing is defined by Jurowiecki [21] stated in the book Wisdom of Crowds [22]. The principles are illustrated as: diversity of opinion, independence, decentralization, and aggregation of opinions. Diversity of opinion describes that every crowd worker should have his/her private information. Independence requires that workers' opinions should be independent from each other. Decentralization requests that workers can specialize and give feedback based on local knowledge. Finally, aggregation means that some mechanism should exist to combine individual opinions into a collective decision.

To explain how crowdsourcing works, Buecheler et al. [9] used the framework proposed by Malone et al. [1] to map the different elements of a crowdsourcing task to 4 basic "questions": Who, Why, What, and How. Figure 1.1(a) shows this framework. It represents two pairs of related questions: 1) who is performing the task? Why are they doing it? And 2) What is being accomplished? How is it being done? The first pair of questions relate to the subject of crowdsourcing by identifying who will be responsible for the task; whereas the second pair of questions relate to the object in crowdsourcing which will be done by subject. The term "gene" is proposed to work as a particular answer to each of the questions. Figure 1.1(b) gives the potential answers (genes) to the questions:

- For the question "Who", there are two basic genes: crowd and hierarchy. Hierarchy represents a



(a) Crowdsourcing genes

Who
- Crowd, Hierarchy

Why
- Money, Love, Glory

What, How
- Create
  - Collection, Contest, Collaboration
- Decide
  - Group Decision
    - Voting, Averaging, Consensus, Prediction Market
  - Individual Decisions
    - Market, Social network

(b) Description of the genes

Figure 1.1: Framework to integrate elements of crowdsourcing [1, 9]

traditional hierarchical organizations; the question is answered when someone assigns a particular person or group of people to perform the task. This traditional way of solving problems is not a typical crowdsourcing task. However, peer reviewing, which is also deemed as one way of crowdsourcing, is likely to fall into this type. In a peer reviewing task, students are assigned by the instructors to grade their classmates′ assignments, in which same group of students are involved in finishing the assignments and accomplishing the grading task. Crowd is another gene which means tasks can be completed by anyone in a large group of people who choose to do so, without being assigned by anyone. That means the answers for the tasks are provided by volunteers.

- For "Why", three genes are provided: 1) Money, which means the financial gain as the motivation for anyone who accomplishes the task. 2) Love gene could take different forms, such as their intrinsic enjoyment for doing the task. 3) Glory could be the motivation of being recognized by peers for their contributions.

- "What" has two basic genes: create and decide. Create refers to creative crowdsourcing tasks, which requires participants to generate something new. For instance, a graphical design for a T-shirt, or writing a paragraph of an article is a typical creative crowdsourcing task. Decide represents the predefined solutions which presented to the participants as answers choices to the proposed task. Workers are required to evaluate and select alternatives to the task, such as evaluating the quality of images by choosing answers from provided scales. There are actually many more answers for the goal of crowdsourcing including collect (e.g. collect facts for news articles from crowdsourcing), and funding (funding projects by multiple of people), etc.

- Answers for "How" are related to the goal of the task, which is "What". As such, the illustration of "What" and "How" in Figure 1.1(b) are combined with each other. If the task is to create

TABLE 1.1

Editing existing Wikipedia articles as example to illustrate the framework consist of four questions (What, Who, Why, How) [1]

Example	What		Who	Why	How
Edit existing Wikipedia articles	Create	New version of article	Crowd	Love, Glory	Collaboration
	Decide	Whether to keep current version	Crowd	Love, Glory	Consensus

something new ("What"), then collection, contest or collaboration ("How") can be the processes associated to it. Collection occurs when the answers contributed by members of the crowd are independent from each other. Videos on YouTube are created independently, and it is a typical example of collection crowdsourcing process. Contest is to select the best entries from a collection of answers. The collaboration occurs when the crowd workers work together to create something. For example, multiple workers create one paragraph each to compose an article. For decide tasks, there are two possible genes: group decision and individual genes. Group decision refers to the process of generating answers by assembling the decisions from a group of workers. However, individual decision represents the crowdsourcing process where a singular person makes a decision that can be informed by crowd input, it does not need to be identical for all.

In order to give a better understanding of the framework, editing existing Wikipedia articles is illustrated as an example in Table 1.1. Depending on the purpose of the task, the way to reach the goal is different. Consider for example, if the task is to create a new version of one Wikipedia article, collaboration can then be utilized as the approach, to accomplish this task by the crowd workers. The reward for the crowds could be either love or glory. While the task is to decide whether to keep current version of the Wikipedia article, then the decision is made via consensus.

Since Amazon launched Mechanical Turk (MTurk) in 2005, crowdsourcing has become a powerful mechanism for completing large scale tasks which involves human intelligence. Human intelligence tasks (HITs) were first popularized by MTurk [23], and they represent single, self-contained tasks. HITs are usually easy for human beings to complete (through crowdsourcing), however, either take large amount of time and effort, or are impossible for computers to accomplish. That means, humans can outperform computers in these types of simple tasks. Some of the common examples of HITs are:

(1) transcription: such as transcribe the radio/media into text. As an example, standard rates in

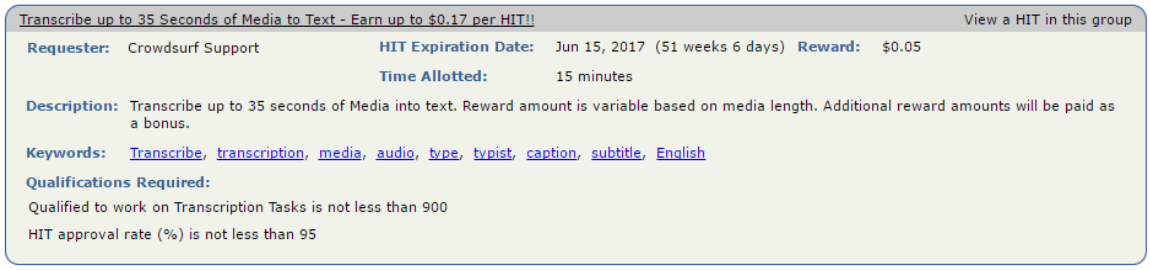


Figure 1.2: Example of HIT posted on Amazon Mechanical Turk[2]

the medical transcription industry are $100 for every 1,000 lines of text, or 10 cents per 65 character line, with a 98% average accuracy rate[1]. However, in [24], the author claimed that by using the SMS server they developed, which is based on crowdsourcing application, the cost can be dropped to 2 cents per line.

(2) Software localization: It is very common that multiple languages exist in one area or country, and translation of them would be very difficult. For instance, there are over 60 distinct languages in Kenya. So it is impossible for software companies to include all these languages into their software interface, since no translation service exists [24]. However, by crowdsourcing the translation tasks, it is possible to localize the language used in the software.

(3) Surveys and market research: conducting large demographic surveys will be much easier and cheaper when crowdsourcing those as HITs, such as collecting product review through MTurk.

In addition to the examples listed, there are many other HITs that have gained success through crowdsourcing such as filtering articles or images based on requirements, sentiment translation in tweets or reviews, and data extraction, etc. The features of these types of tasks have been limited to those that are low in complexity, independent, and require little time and cognitive effort to complete [25]. Markets such as MTurk are places where anyone can post tasks (HITs) to be completed and specify reward (prices) for completing them [26, 27]. Figure 1.2 gives an example of one of the HITs posted on MTurk. As shown in the figure, the requesters post the media transcription task on MTurk. They will need to give detailed description of the tasks, and also very importantly, list the reward for completing the tasks. As shown in Figure 1.2, the description is given as "Transcribe up to 35 seconds of Media into text. Reward amount is variable based on media length". The reward for transcribing each Media is up to $0.17. People who are interested in doing this transcription task and fulfill the required qualification could try to do the work. Once the task is finished, workers will

[1] Official Newsletter of the Medical Transcription Industry Association of the Philippines, 2008.
[2] https://www.mturk.com/mturk/findhits?match=false

TABLE 1.2

List of platforms for crowdsourcing [2]

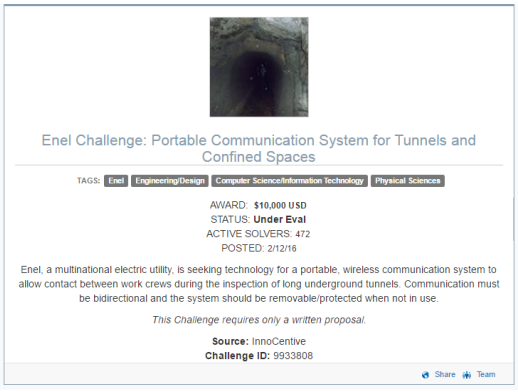
Markets	Purpose	Launch	Reward
OpenStreetMap	Geographic content	University College London, 2004	None
TxtEagle	Text translations	Start-up, 2009	Micro-payment(<$1)
Wilogo	Graphical design	Start-up, 2006	Remunerated (approx. €300)
InnoCentive	Problem solving and innovative projects	Eli Lilly, 2001	Remunerated (>$1 000)

be given the claimed reward. In March 2007, Amazon claimed that the MTurk users has reached over 100,000 from over 100 countries [26].

In addition to MTurk, there are many other platforms nowadays for crowdsourcing tasks. Table 1.2 gives a list as examples of these markets [2]. OpenStreetMap (OSM) was inspired by the success of Wikipedia, which is an Internet encyclopedia and those who use Wikipedia can edit almost any article accessible [28], and the preponderance of proprietary map data in the UK and elsewhere [29, 30]. It is a collaborative project to create a free editable map of the world. Similar to MTurk, TxtEagle also allows requesters to post the HITs and crowdsource the tasks. However, the difference is that Amazon MTurk is based in personal computer, whereas TxtEagle is a mobile phone-based system, which enables people to complete simple tasks on their mobile phone [24]. The reward/payment of TxtEagle is also quite similar to MTurk. Wilogo and InnoCentive pay much better than MTurk and TxtEagle since the tasks posted there involve higher worker expertise and question difficulty. Wilogo allows companies to source design with speed and efficiency. For example, the design of logos, web sites and brochures. Whereas InnoCentive is a crowdsourcing company that accepts commission research and development problems in engineering, computer science, chemistry, math, and business, etc. Figure 1.3 gives examples of the tasks posted on InnoCentive.

Different from traditional way of solving problems through employees or experts, crowdsourcing involves participants with different level of expertise and knowledge, thus quality risk is a big problem for tasks through crowdsourcing. The cost and time for verifying the correctness of the answers submitted by crowdsourced workers might significantly exceed the cost and time for performing the task itself [31]. A common solution is by using redundancy: the same task is completed by multiple workers. Then aggregating the answers to get the final consensus answer according to different principles, such as majority voting or expectation maximization, etc.

[3]https://www.innocentive.com/ar/challenge/browse



(a) Challenging problem in computer science

(b) Challenging problem in engineering

Figure 1.3: Example of tasks posted on InnoCentive [3]

Crowdsourcing has claimed its success in many areas and applications [22, 32], the benefits for solving problems via crowdsourcing include:

(1) Come up with new ideas, such as the web-based t-shirt company called Threadless that crowdsources the design process for their t-shirts. Anyone could submit a graphic design for t-shirt to the company, if the design got selected, it will be printed and made available for sale. It was reported the company gained great success with 60,000 t-shirts sold in June 2006. The new ideas for graphic design from crowds are the main factor for the achievement of the company.

(2) Lower the costs. As an example, data collected by Zaidan and Callison-Burch [33] through MTurk is shown here. They hired Turkers (workers on MTurk) to translate 1792 Urdu sentences. Each Urdu sentence was translated by 4 non-professional translators. At the same time, 4 professional translators were also hired for translation. The cost of non-professional translation was $0.10 per sentence. While the cost of professional translation to be around $6 per sentence. As a result, crowdsourcing lower the cost greatly.

(3) Shorten the time spent on tasks such as crowdsourcing the grading process would save a lot of time for the instructors. While grading by instructors, the process is sequential since instructor need to grade the submissions one by one. However, crowdsourcing makes the process more like a parallel system hundreds of thousands of graders online are able to grade at the same time slot. Thus shorten the time spend on finishing the task.

Despite the promises, challenges remain. One of the biggest challenges is we do not have the control of the quality of workers for crowdsourcing. The workers could be genuine experts, biased workers, or malicious workers. Some other challenges include how to recruit and retain the workers [22,34,35],

how to combine the contributions of the workers [36] and how to evaluate them [37] are still under researching and discussion.

1.2 Challenges of labeling through crowdsourcing

Crowdsourcing offers a cost-effective and scalable way to obtain labels by utilizing human computation via the Internet. However, the quality of the work performed by the crowd can greatly vary across individuals, which risks compromising the overall data quality. Due to the fact that the typical crowdsourced tasks are tedious and the reward is small, errors are common even among workers who make an effort [38]. In extreme cases, some workers might submit arbitrary answers independent from the questions in order to collect their reward fee. This rapid, decontextualized, and many times anonymous work through crowdsourcing systems such as Mechanical Turk often yields low-quality results. Some common methods have been proposed to increase the work quality:

(a) Increasing the payment for each task completed. Paying more can improve individual performance, but it actually reduces quality after a certain point because it incentivizes speed [39].

(b) Worker filtering [26, 40]. For example, identifying the poor quality workers and excluding them to get better results.

(c) Using redundancy to obtain labels from multiple workers for a given question, which is called repeated labeling, and then aggregating the labels to get a single consensus label. It is the most widely used method to tackle the noisy data obtained through crowdsourcing. A common approach is simple Majority Voting (MV) [41–44], which is easy to use and can often achieve relatively good empirical results depending on the accuracy of the worker involved.

1.2.1 Inferring true labels from repeated labeling

Repetitive labeling makes an effort to improve the consensus results via achieving labels from multiple workers for a given item. To aggregate the labels, many different approaches have been developed for the purpose. Majority voting assigns the label which receives the maximum number of votes as the aggregated answer. If ties exist, a label is selected randomly as the final result. In MV method, the trustworthiness of every worker is assumed the same as each other. As discussed in many existing research, the quality of labels provided by workers actually are varied greatly. It is necessary to take into account this factor. An alternative is use weighted voting [38, 45], in which the labels from a worker is weighted in proportion to his trustworthiness or accuracy, which

defines the quality of the work performed by a labeler. The major problem here is in many cases, the true labels usually are not available. The missing of truths makes it difficult to evaluate the worker's performance, which is important to obtain the aggregated labels in weighted voting. To deal with the challenge, a variety of work has investigated means for assessing the quality of worker labels or the difficulty of labeling tasks. As an example, Dawid and Skene [46] utilized Expectation Maximization (EM) algorithm to learn each worker's error rate by a latent confusion matrix, and at the same time, estimate the true labels for each item. Whitehill et al. [45] proposed a methodology to estimate the label by taking into account both worker's performance and the difficulty of the labeling task.

With the gold data, which is the labels obtained from experts, it is easier to evaluate the performance of the work of crowd, by comparing their labels to those of the experts. As a result, more accurate aggregated results consensus labels could be achieved. Snow et al. [42] adopted a fully-supervised Nave Bayes (NB) method to estimate the consensus labels from such gold labels. Compared to the algorithms proposed in [46] and [45], which with the hypothesis of gold data is not available, the methodology in [42] assumes the expert labels are known. The approaches under the expert supervision (with gold labels known), are supervised learning for inferring true/consensus labels from crowdsourced data. Estimating consensus labels without the availability of truths is formulated as an unsupervised learning problem. Due to the fact that full-supervision can be costly in expert labeling, which is the reason crowdsourcing is proposed in the first place, it is usually unrealistic to get the consensus labels through supervised leaning method.

Although voluminous amount of expert data cannot be expected, it is possible to obtain a proportion of gold data from experts. To balance the consensus accuracy and labeling cost, a lot of research has been dedicated to semi-supervised learning and active learning [44, 47, 48]. In semi-supervised setting, a small number of gold labels are known, and the estimated consensus labels can be obtained based on the available information. Active learning, on the other hand, optimally picks the item for which the corresponding predictions are the most ambivalent, or least certain, to be labeled by the workers [48].

1.2.2 Filtering workers from the crowd

Since a major drawback of crowdsourcing systems is that we have no control over the quality of the crowd, there might be workers who try to maximize their financial gains by generating generic labels rather than actually working on the task [49]. Many previous work showed that workers

have a tendency to make significant error in crowdsourcing tasks [41, 49, 50]. It poses the problems such as the unpredictability of workers (e.g. reliability, expertise), and spam or malicious behaviors while labeling tasks. To tackle this, worker filtering techniques are proposed to remove the poor performance quality labelers.

Setting up control questions is a common way for crowd filtering. For example, first test the workers′ performance by asking them to complete a set of predefined questions for which true labels are already known. Only the workers with desirable accuracy are then allowed to do the actual labeling task. It is also possible to design the task in such a way that it becomes less attractive to workers who provide generic arbitrary labels. Eickhoff and de Vries [51] concluded that malicious workers are less frequently encountered in novel tasks that involve a degree of creativity and abstraction.

However, the control questions are not always applicable in many crowdsourcing systems. Some other solutions are proposed to deal with the problem. Such as the approaches as simple as looking at the time spent per task [26, 52], to complex probabilistic models to check trustworthiness of workers [50, 53]. These methods dedicated to discover different characteristics between reliable workers and undesirable workers, and filter the crowd based on these features.

1.3 Labeling complex tasks through crowdsourcing

In the last decade crowdsourcing has become increasingly popular [44], and it is now in widespread use for data-processing tasks such as image classification, video annotation, translation, and recommendation [38]. As discussed in Section 1.2, the data obtained through the crowdsourcing systems might be highly noisy, especially for complex tasks. Different from the simple tasks accomplished through MTurk, which are low in complexity, independent, and require little time and cognitive effort to complete [12], complex tasks require the workers possess knowledge or skills in specialized task domains to solve the problems. Due to the feature of online crowd workers′ diverse background and education, it is typical that these workers do not have the required skills to complete the labeling tasks, thus the quality of crowdsourced results cannot be guaranteed. To combat this, some crowd systems break down work into simpler tasks (sub-tasks), and get feedback for the sub-tasks via crowdsourcing [12, 54]. As an example, Kittur et al [12] gave writing a short article about a newspaper, a travel guide or encyclopedia as a decomposable complex task in their work. The task could be decomposed into sub-tasks such as deciding on scope of the article, collecting relevant information, taking pictures, and laying out the document. Each of the sub-tasks could be

completed by separate sets of workers.

Another way to improve the work performance of crowd for complex tasks is providing rubrics to workers [4, 14]. By offering the rubrics, workers would be able to develop a better understanding of the goals and the standards of a task. It further helps labelers track how they are progressing towards those goals [55]. Yuan et al. [4] investigated the effects of expert rubrics in the domain of design critique, and they provided evidence that an expert rubric helps novice feedback providers produce similar values as expert.

Defining rubrics is especially useful in the grading process, in which evaluating submissions from students as a labeling task. Consider for example, grading students′ essay or research paper from a data mining class is a complex task which is different from typical MTurk problems. The traditional way of providing grades and comments as feedback to students in a school is via the evaluation from instructors or teachers. However, it has become a big challenge for large scale classes [7, 56] such as Massive Open Online Classes (MOOCs), which are distributed on platforms such as Udacity, Coursera and EdX [5]. In addition, the grading processes are really time consuming and tedious. As an example, there were around 100,000 students signed up for the online machine learning course offered in Coursera platform. To evaluate students′ work, a scalable way for grading is required for such large scale classes. Due to the restriction of un-scalability of traditional ways, grading through crowdsourcing has been developed as one of the prevalent ways to solve the problem. Different from the decomposable complex tasks as discussed in [12] and [54], evaluating an essay in most cases needs to take into account the context of the whole article (un-decomposable). For example, literature review is an important section in a research article. If we separate this part as one sub-task of the article, it is possible that a review which is unrelated to the essay topic would receive a relatively high score, which is undesired. Thus providing rubrics is more appropriate way to help the crowd improve their work performance instead of decomposing the labeling task into smaller sub-tasks in this case. We call this type of task as un-decomposable complex task.

Evaluating images is another example of un-decomposable task which demonstrates the plausibility of utilizing rubrics. While evaluating the quality of the image, it is not reasonable to decompose the images into several parts, and judge the quality based only on the parts. Figure 1.4 shows an example while evaluating the image quality. If a scale from 0 to 5 is provided for crowdsourcing task as evaluating the quality of images, with 0 means “Not clear at all” and 5 as “Very clear image”. Figure 1.4(a) presents the original image, whereas 1.4(b) and 1.4(c) are different parts of the original image after decomposition. Workers who evaluated part 1 probably give 0 score,

(a) Original image (b) Part 1 of original image (c) Part 2 of original image

Figure 1.4: Image quality evaluation as example for un-decomposable complex task[4]

while those who graded part 2 might give 5. However, neither of them provide meaningful results, since their evaluation are only based on partial aspects of the image.

In order to overcome the problem of insufficient domain knowledge and expertise of the crowd workers, a lot of research has been done to propose various methodologies, in addition to decompose the work into easier sub-tasks and define rubrics. One of the popular approaches for grading large scale classes is called peer reviewing, in which students are responsible for grading assignment submissions of their classmates. Thus the same group of students are involved in completing the assignments and in the process of grading. The advantages of peer reviewing are that it is a scalable way for grading large scale classes since the same amount of students will be available for grading as the number of submissions, and the reward could be flexible, not necessarily monetary driven. For instance, some of the research proposed that students′ review effort could be taken into account as a component of their final grades [7]. However, there exists disadvantage at the same time. Consider the situation that students might give bad grades for all the submissions he/she is responsible for reviewing to make his/her own grade stand out. This case disqualifies the peer reviewing as useful grading process.

Although crowdsourcing provides a scalable way for labeling/grading large scale classes, the cost might still be a problem, even though it has lowered the cost significantly compared to labeling through experts. To further reduce the cost for labeling, researchers at Stanford recently proposed a semi-automatic method for grading [57]. This approach is specifically designed for grading students′ submissions from class. It was done through organizing the "space of possible solutions", instructors would provide feedback for a few submissions, and their feedback could be propagated to the rest. Figure 1.5 illustrates the grading process of this method. First, all the assignment submissions are clustered by the researchers. The small circles, triangles, and squares in the figure represent the

[4]https://www.crowdflower.com/use-cases/

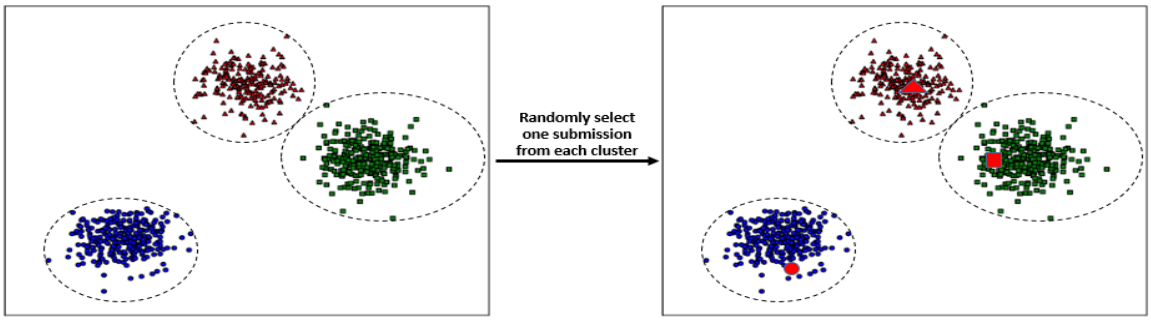


Figure 1.5: Graphical illustration of semi-automatic grading method

submissions, and they are assigned into three different clusters (with large oval surrounded). For example, while grading the computer programming assignment submissions, the features used while clustering are: unit test results and abstract syntax tree (a tree representation of code structure) and tree edit distance [57–59].This method assumes all the submissions in the same cluster are similar enough. After clustering the submissions, one or more submissions are selected randomly from each cluster and graded by the instructor. As shown in Figure 1.5, the enlarged circle, square and triangle, are the selected submissions for grading. Finally, propagate the given grades to the rest submissions for each cluster respectively. Although researchers proposed the possibility of utilizing the methodology, it has not yet been used for grading. However, some instructors have used it to highlight common mistakes. Other problem with this method is finding features of the submissions for clustering sometimes could be really difficult. For example, what should be used to cluster research papers in philosophy or history studying? And how could this approach be generalized into different types of labeling tasks? For example, how to extract features for clustering in an image labeling task?

1.4 Formalization of basic concepts used in crowdsourcing systems

There are some concepts used in our research, and they are defined as follows:

Gold truth/label The label which could reflect the true quality of one item that needs to be labeled.

Observed label Observed labels are the answers obtained from the crowd workers for the items which need to be labeled.

Consensus label The label obtained after aggregating the multiple observed labels using redundancy technique, which collects labels from multiple workers for a given item in order to improve

the work quality of the online crowd.

Experts Those who have enough knowledge and ability to give best understanding of the labeling task. For example, instructors, who specified and assigned the complex assignment for students, are considered as experts. We assume the labels offered by experts are the closest answers to gold truths. Thus in our research, we use the expert labels as the gold data.

Complex task The labeling task which requires the workers possess knowledge or skills in specified task domains [55]. As an example, grading a research essay can be viewed as a complex task.

Decomposable complex task One type of complex tasks for which it is possible to break down the task into smaller and easier sub-tasks. The sub-tasks can then be completed by separate groups of workers.

Un-decomposable complex task The complex tasks which need to be labeled within the context of the to-be labeled item. Providing rubrics for the task is a more appropriate way to improve the label quality compared to breaking it down into sub-tasks (as discussed in Section 1.3).

Simple task Tasks which are low in complexity, independent, and require little time and cognitive effort to complete [12]. Simple tasks are able to be posted and completed on markets such as MTurk. For instance, labeling images with keywords, transcribing radios into text, and rating the layout of web pages are simple tasks.

Crowd scoring task The labeling tasks with possible label set consists of either ordinal categorical or numerical labels.

1.5 Dissertation overview and major contributions

In this dissertation, three different approaches are presented for dealing with unreliable workers in crowdsourcing systems, and we focus on improving the quality of consensus labels in complex labeling tasks. The research in this thesis concentrates on the tasks where the possible label set consists of ordinal labels or numerical labels. The presented work separates the workers into different categories and filters the spam workers, who provides generic arbitrary labels for an item. At the same time, the proposed approaches detect the biased workers, who present the tendency of providing higher (or lower) feedback than the truths. By correcting the labels obtained from biased workers, it is possible to obtain better results. To deal with the limitations of crowdsourcing systems for complex tasks, such as high worker variance (high uncertainty) due to insufficient task skills, we

then propose a multi-view approach to provide uniformity among the workers, as well as guide the crowd towards the goal of the labeling task. Although it is unrealistic to have expert gold labels for all the items need to be labeled, a small set of gold truths is usually possible to achieve. As such, a semi-supervised learning methodology is proposed to filter the unreliable workers by utilizing limited gold labels. The detailed layout of the thesis is presented next.

Chapter 2 presents the background related work on the labeling through crowdsourcing. Two types of aggregating algorithms in order to obtain consensus labels from repeated labeling are discussed: *Non-iterative* methods such as MV to get a single consensus label. *Iterative* methods, which performs a series of iterations until converge to obtain the estimated labels for each item. Literature review of existing approaches for both bias analysis and spam worker filtering is presented in this chapter. This chapter also gives the survey of the methodologies to perform complex labeling tasks through crowdsourcing, and show a set of crowdsourcing datasets utilized for existing research in literature.

In *Chapter 3*, a framework combines spam removing algorithm and de-biasing method is presented to improve the quality of consensus labels. The Spam Removing and De-biasing Framework (SRDF) adopts and adjusts the spam worker filtering algorithm from [60] to detect and remove the unreliable workers which provide generic labels for items. It then uses the de-biasing method developed in this research to detect the workers with biased behaviors, and then correcting the labels offered by them. Demonstration that fewer workers are required to achieve desirable results after using SRDF approach, is also presented.

The framework developed in Chapter 3 mainly aims at numerical labeling tasks such as grading for students submissions from a class. It is also interesting to investigate labeling tasks with possible label set with categorical ordinal data. Consider for example, relevance judgement for a set of (query, website) pairs data. In addition, the worker categorization in Chapter 3 adopted from [60] is somewhat overlapping with each other. Based on these limitations, *Chapter 4* gives a novel way to cluster the crowd workers, and presents an optimization based truth discovery method, which is applicable for both numerical and categorical labeling tasks, to estimate the consensus labels from observed data. The proposed Iterative Self Correcting Truth Discovery (ITSC-TD) methodology not only removes the spam workers, it also deals with the sloppiness in a crowd scoring task. Sloppiness, which represents the phenomena of observed labels which fluctuate around the true labels, is one type of error that has rarely been discussed in research. We show that sloppy workers with biases, who constantly give higher (or lower) answers compared to true labels, can be utilized to improve

the quality of estimated labels.

As mentioned in Section 1.3, crowd workers usually do not possess the skills in specified task domains to perform the complex labeling work. *Chapter 5* concentrates on improving the work performance of labelers in complex labeling tasks through a multi-view approach. The multi-view approach use each rubric criteria as one view, and investigate the worker reliability, which defines the worker's trustworthiness, for this view. The methodology proposed in Chapter 4 is utilized to obtain high quality consensus labels for each view of the item. Finally, the consensus view labels are used to achieve a single overall label for each item.

All the work presented in previous chapters (Chapter 3, 4 and 5) is conducted under unsupervised settings, in which we assume no gold data is available while estimating the true labels. We give hypothesis that a subset of gold truths is available in *Chapter 6*, and develop a novel semi-supervised worker filtering algorithm to block noisy workers, before estimating the true labels from observed data. The small set of items with gold labels known, together with the remaining items, are utilized to filter out the spam workers in order to improve the consensus results. The research conclusion and potential future work, is presented in *Chapter 7*.

The major contributions of the dissertation are:

- Most existing work does not distinguish biased workers from the spam workers. In contrast, the research in this thesis proposes to recognize these two types of workers separately, and demonstrates that labels obtained from biased workers are useful in inferring the true labels from observed data. A framework combines spam worker filtering and bias worker correcting is proposed to improve the consensus results.

- A hierarchical categorization for different types of crowd workers is presented for our research. Based on the categorization, a framework which is applicable for both numerical and categorical ordinal data is proposed to mitigate the sloppiness in crowd scoring tasks.

- A multi-view approach, which can investigate worker's reliability on each labeling rubric for complex labeling tasks. Real world data is collected and utilized to demonstrate the effectiveness of the methodology.

- Semi-supervised worker filtering – truth discovery framework, which utilizes a small set of gold labels to eliminate unreliable workers, for estimating high quality consensus labels from noisy crowdsourced data, is proposed.

CHAPTER 2

REVIEW OF WORK ON CROWDSOURCING SYSTEMS

Crowdsourcing, defined by Jeff Howe as "an idea of outsourcing a task that is traditionally performed by an employee to a large group of people in the form of an open call" [61], has played important roles in data mining activities. Especially with the explosive growth and widespread accessibility of the Internet, a lot of research has been done in crowdsourcing. Quality control on crowdsourcing systems, which makes efforts to obtain high quality labels, is one of the hot topics among them [31, 41, 42, 62]. There are two main reasons which account the importance of quality control: a) Nowadays, the overwhelming majority of workers are attracted by the financial reward [51], and b) The worker may lack expertise or skills to handle some kind of complex job [4,63]. Some other reasons could be misunderstanding the tasks, personal preference, or fatigue from long working, etc.

Repeated labeling [41], which requests multiple labels for a given item from several different workers, is a most widely used approach to estimate true labels of the item. To get a single consensus label, aggregating algorithms is required to combine the multiple observed labels. Another popular way to improve the quality of output labels is filtering the workers with poor performance on the labeling task [26, 40]. It requires to study the worker reliability through the observed labels, which defines the trustworthiness of the worker.

This chapter presents related work in the field of aggregating algorithms for labels obtained via repeated labeling, and worker filtering, which entails spam worker recognition and removing, as well as bias detection. Section 2.1 shows works related to aggregating approaches. Works related to worker filtering are presented in Section 2.2. Also, we give literature review of methodologies developed to improve qualities in complex labeling tasks, which is given in Section 2.3. A brief overview of crowdsourcing data sets used in current research are presented in Section 2.4. Finally, Section 2.5 gives the chapter summary.

2.1 Related work on aggregating algorithms

In order to get expert-wise answers, redundancy is usually utilized for crowdsourcing process, which means multiple labels are collected from the crowds to the same item. This raised the challenge of how to aggregate the labels into a single, accurate answer. A lot of research has been done for the label aggregation problem. In general, the aggregation techniques could be classified into two categories [64]:

(1) Non-iterative: uses heuristics to compute a single aggregated value of each item. Majority Voting/Decision [65], HoneyPot [66], and ELICE [67].

(2) Iterative: performs a series of iterations until converge. Each iteration consists of two updating steps: a) updates the aggregated value of each item based on the expertise of the workers from the crowds; b) adjusts the expertise of each worker based on the labels given by her. Expectation Maximization (EM) [45, 46, 68–72], GLAD [45], ITER [38], Gibbs sampling [73] are iterative aggregating algorithms.

Before giving details about how these algorithms work, we would like to give explanations of several concepts: worker expertise, item difficulty, and test questions [65].

Worker expertise – the ability to capture the behavior of a worker such as the accuracy or reliability of the worker.

Item difficulty – the ability to measure how difficult is the item which needs to be labeled. For example, items are completed by a same set of reliable workers, the item with less workers labeled correctly is more difficult and thus the item difficulty is higher.

Test questions/Gold Set the set of items whose labels are known before-hand. It is mainly used to filter spam workers and initialize the expertise of the workers. CrowdFlower uses this approach to measure the quality of the workers. For example, if there is a task to rate the sentiment of a tweet, then the test question could be shown in Figure 2.1. A sentiment rating question is set and workers are required to provide answers for the question. Assume both highly positive and slightly positive are enabled as correct answers, then workers who get the test question correct are selected.

[1] https://success.crowdflower.com/hc/en-us/articles/202702955-Guide-to-Creation-Page

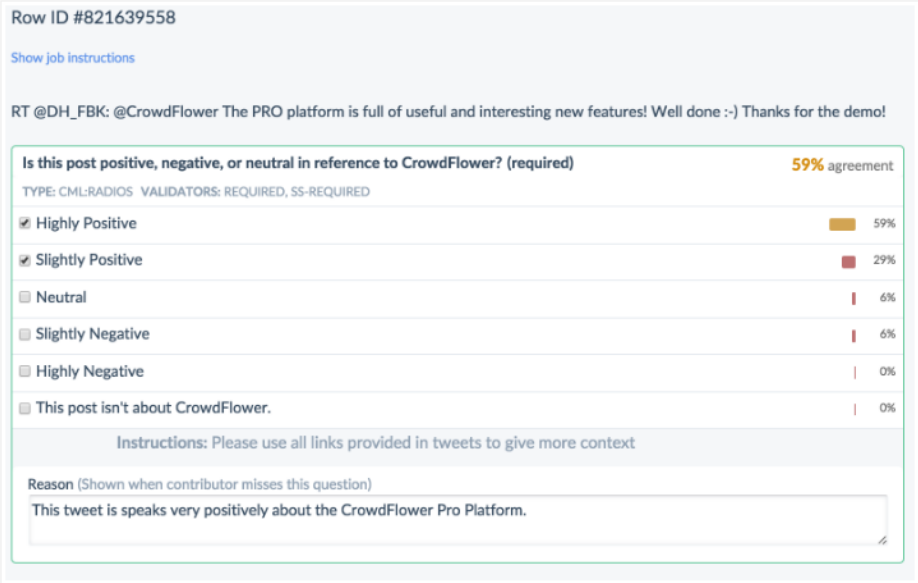


Figure 2.1: Example of test question (gold set) while rating the sentiment of a Tweet in CrowdFlower[1]

2.1.1 Non-iterative algorithms

Majority Decision/Voting [65] is a common and simple non-iterative approach in which the answer with the highest votes is selected as the final aggregated value. As an example, if there are totally 8 people working on an item in labeling task which label should be either A or B (but not both A and B). Suppose 5 of them voted for A, 3 voted for B, then the aggregated answer is A, since $5/8 > 3/8$. There are also methods such as average approach, which average all the voted answers into one final value, as non-iterative algorithm for problems with numerical data as answer. Whereas majority voting usually is used for categorical case, such as classification or relevance judgement problem. The problem with majority voting is that it does not take into account the expertise and reliability of the workers.

HoneyPot (HP) [64, 66] works similar as majority voting, the difference is HP filters the untrustworthy workers in a preprocessing step. In this step, HP provides a set of test questions and inserts into original questions randomly. Workers who fail to answer a specified number of test questions are rejected as spammers. After filtering, the final label is obtained by using majority voting among the remaining workers. Compared to majority voting, HP considers the spam problem to some extent. However, one of the very important disadvantages of HP is test question set is not always available, and it will take extra cost to do it.

ELICE stands for expert label injected crowd estimation [64, 67], and it is an extension of

HP. ELICE also uses the set of test questions, but it also estimates the expertise of workers by measuring the ratio of his answers which are identical to true labels. It then calculate the difficulty level of each item by the expected number of workers who correctly label a specified number of test questions [64]. ELICE also uses test questions, and the problem of unavailability of these questions might also exist sometimes.

2.1.2 Iterative algorithms

Expectation maximization (EM) [45,46,70–72] is very commonly used aggregation technique, and many researches have extended the algorithm to specific crowdsourcing problems. EM solves a maximum likelihood problem of the form:

$$l(\theta) = \max_{\theta} \log \int p(x, z; \theta) dx \tag{2.1}$$

where θ are the parameters of the probabilistic model we try to find (e.g. workers' expertise or reliability), x are the unobserved variables or missing values (e.g. true labels), and z are labels collected from crowds. Assume Q is the probability distribution over the x's, then EM algorithm iterates the following two steps until converge: 1) *E(Expectation)-step*: Compute $Q(x) = p(x|z;\theta)$, which is using current parameters (θ) and observations (z) to estimate unobserved variables (x). 2) *M (Maximization)-step*: Compute $\theta = \underset{\theta}{argmax} \int q(x) \log p(x, z; \theta) dx$, which uses estimated unobserved variables and observation to re-estimate maximum likelihood of the parameters.

As an example, giving a relevance judgement problem with choices (labels) set G as $\{0, 1\}$, number of workers M is 2. There are totally $N = 2$ documents. Assume each documents should be evaluated by 2 workers. The grades given are: $z_{11} = 0, z_{21} = 1; z_{12} = 1, z_{22} = 1$. We apply EM algorithm here, in which π^j_{kl} and p_k are treated as parameters and R_{ik} as missing values. All the notations used in the process and their definitions are shown in Table 2.1. The EM algorithm involves the following steps:

(1) Initialize the true relevance values of each document: $R = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

(2) Given the current estimate of R_{ik}, calculate the maximum likelihood estimates of π^j_{kl} and p_k:

$$\hat{\pi}^j_{kl} = \frac{\sum_{i=1}^{N} R_{ik} n^j_{il}}{\sum_{l=0}^{G} \sum_{i=1}^{N} R_{ik} n^j_{il}}, \quad k \in \{0, 1\}, \quad l \in \{0, 1\}, \quad and \quad j \in \{1, 2\} \tag{2.2}$$

TABLE 2.1

The definition of notations in the example of using EM algorithm for document relevance judgment task

Notation	Definition
M	Number of workers
N	Number of documents need to be evaluated
G	Label set ($\{0, 1, ..., k\}$)
p_k	Probability of a document has true relevance label of k
$R_{ik} (\in \{0, 1\})$	True relevance value of document i is k if $R_{ik} = 1$
R	Matrix with R_{ik} as element
z_{ij}	Label worker j given to document i
π_{kl}^j	Probability of worker j provides value l given that k is the true relevance value of a document
n_{il}^j	Number of times worker j provides grade l for document i

$$\hat{p}_k = \frac{\sum_{i=1}^{N} R_{ik}}{N}, \quad k \in \{0, 1\} \tag{2.3}$$

For instance, while $k = 0, l = 0, j = 1, \hat{\pi}_{0,0}^1 = \frac{1\times1+0\times0}{1\times1+0\times0+1\times0+0\times1} = 1, \hat{p}_0 = \frac{1+0}{2} = 0.5$. Similarly, a set of $\hat{\pi}_{kl}^j$ and $\hat{p}_k$ could be calculated for different values of k, l and j.

(3) Calculate the new estimate of R_{ig} ($g \in G$) based on $\hat{\pi}_{kl}^j$ and $\hat{p}_k$ as:

$$Pr(R_{ig} = 1 | n_{i0}^j, ..., n_{iK}^j; \pi_{g0}^j, ..., \pi_{gK}^j) = \frac{p_g \prod_{j=1}^{M} \prod_{l=0}^{K} (\pi_{gl}^j)^{n_{il}^j}}{\sum_{k=0}^{K} p_k \prod_{j=1}^{M} \prod_{l=0}^{G} (\pi_{kl}^j)^{n_{il}^j}}, i \in \{1, 2\}, g \in \{0, 1\} \tag{2.4}$$

when $i = 1$, and $g = 0$, $Pr(R_{ig} = 1 | n_{i0}^j, ..., n_{iK}^j; \pi_{g0}^j, ..., \pi_{gK}^j) = 0.5 \times 1^1 \times 0^0 \times 0^0 \times 1^1 \div (0.5 \times 1^1 \times 0^0 \times 0^0 \times 1^1 + 0.5 \times 0^1 \times 1^0 \times 0^0 \times 1^1) = 1$.

(4) Repeat steps 2 and 3 until converge. Finally, set $R_{ig} = 1$ for the g with the maximum probability as calculated in equation (2.4) for every document i.

EM takes a lot of steps to reach convergence, thus running time could be a critical issue. In addition, EM-based algorithms are highly dependent on initialization parameters and can often get stuck in undesirable local optima [74] (EM is guaranteed to converge to a point with zero gradient). In typical models to which EM is applied, the missing values which need to be estimated are discrete [3].

GLAD [45, 64] is an extension of EM algorithm. GLAD takes into account both worker expertise and item difficulty while aggregating. It is designed to consider two cases: a) if an item is labeled by many workers, then the worker with higher expertise tends to label the item correctly

with higher possibility; b) if a worker labels many items, the item with higher difficulty has a lower possibility to be labeled correctly. The problem with GLAD is that it highly depends on the initial value of worker expertise and item difficulty.

Iterative learning (ITER) [64] is based on standard belief propagation technique [38]. Similar to GLAD, ITER also estimates the worker expertise and item difficulty. The difference is that while GLAD treats the reliability/worker expertise of all labels of one worker as a single value, ITER computes the expertise of each item separately. Thus the expertise of each worker is estimated as the sum of expertise of his labels weighted by the difficulty of associated items. The disadvantage of ITER is the high computation time and different evaluating techniques are required in the same setting.

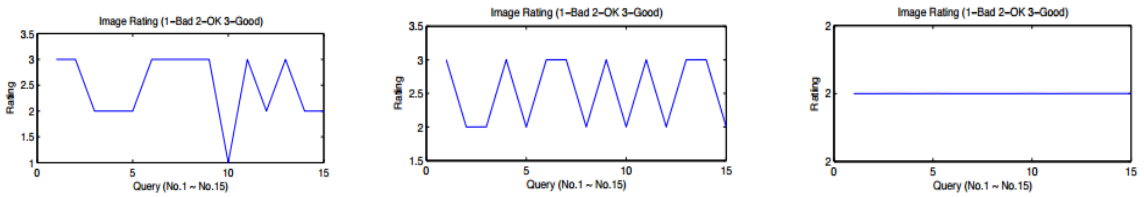
2.2 Related work on worker filtering

On many crowdsourcing systems like Mechanical Turk, people perform short, simple tasks for small amount of money. This rapid, decontextualized, anonymous work provides few motivational incentives and little time for reflection [14,26,31]. Consequently, workers often spend minimal efforts and time to maximize their financial profit, yielding low-quality results. In order to increase the performance of the crowdsourcing system, a variety of research has explored the possibility to filter un-trustworthy workers – spam workers. Different from spam workers, which offers generic arbitrary labels, workers who provide biased responses, however, can be helpful in the process of inferring true labels [75, 76].

2.2.1 Spam detection and removing

Spamming problem during crowdsourcing process has been revealed and reported by many researchers [10, 26, 41, 52, 77–80]. In [26], invalid responses for MTurk tasks are analyzed. It found that MTurk has a number of limitations including the susceptibility of the system to malicious user behavior. Thus the authors proposed special care, such as qualifying users by using automated pre-tests, is needed in formulating tasks in order to harness the capabilities of crowdsourcing approach. To deal with the spamming problem, recent research efforts have been dedicated to detecting and removing spam workers in crowdsourcing systems [10, 52, 77–79].

Authors in [10, 60, 78] categorized the patterns of the workers behavior, and then spam workers are revealed according to the patterns. Zhu and Carterette [10] gave an analysis of the behavior of assessors who participate in preference judgement and different patterns are detected.



(a) Normal assessors (b) Periodic assessors (c) Fixed assessors

Each plot shows the rating an assessor gave to the set of images

Figure 2.2: Patterns discovered for image rating analysis on preference judgment [10]

Unreliable assessors may then be revealed according to patterns analyzed. The authors set up an online survey that asks assessors to give their preferences on variations of the Yahoo! SERP layout for 47 queries formed from 30 topics. In addition, they also ask the assessors to rate the pictures on the layouts. After conducting the study, they plot the time in seconds that assessors spend on each preference judgement against the image ratings. 3 different patterns are shown in image rating analysis. The patterns are presented in Figure 2.2. The normal accessor shown in Fig. 2.2(a) demonstrate no clear pattern in image preference judgements. Periodic assessors in Fig. 2.2(b) presents the tendency to alternate between a subset of ratings in a consistent way. Whereas fixed assessors in Fig. 2.2(c) give every image set the same judgement. The experimental analysis showed that periodic and fixed assessors tend to be cheating, which means assessors present periodic and fixed behavior patterns are likely to be unreliable assessors.

In [60], [78] and [79], workers are classified into different types in crowdsourcing either relevance judgement tasks [60, 79] or software development tasks [78]. Abhinav el al. [78] defined six groups of workers, which are shown in Table 2.2. The first column is the types of workers, and "Description" gives the behavior pattern of the corresponding type of worker. For example, the worker who intentionally gives wrong labels is identified as a biased worker, and who gives random label for any item is defined as a random worker, etc. After defining different types of workers, experiments were conducted to observe how the accuracy of majority voting and expectation maximization algorithm influenced by these groups of worker, respectively. Authors in [60] and [79] classified workers in relevance judgement task into 5 different types of workers: proper worker, random spammer, semi-random spammer, uniform spammer and sloppy worker. The spam removal algorithms based on random error, pattern frequency and precision is then proposed to remove the random, uniform and sloppy spammers separately to increase accuracy.

There are many other ways to remove low quality workers (spammers) without characterizing

TABLE 2.2

Different types of workers in crowdsourcing software development [3]

Type	Description
Expert worker	Expertise in the area with profound domain knowledge and the questions answered correct, with a high probability
Biased worker	Intentionally gives incorrect answers
Random worker	Gives random answers for any question
Uniform worker	With a specific motive give same answers for all the questions
Adversarial Colluded worker	Gives wrong answers by colluding with other,workers having malicious intention
Non-Adversarial Colluded worker	Gives correct answers by colluding with other,workers for the sake of monetary benefits

different groups for crowd workers. Le et al. [77] used a training set to reject unethical workers for relevance categorization tasks in MTurk platform. The training data was used first in an entry training module, in which each worker need to complete 20 query-result pairs relevance judgement before proceeding to test set questions. By using the training questions, the authors estimate worker's accuracy. In [81], an algorithm is proposed to separate the true error rate from biases for MTurk crowdsourcing tasks. By estimating the expected cost for each worker, the author presented that perfect workers will have a cost of zero and spammers will have high expected costs. It is interesting that as long as the errors of the worker are predictable and reversible, which represent bias behavior, the worker is still assigned a low expected cost in this algorithm. Thus, a worker with true errors can be separated and rejected, whereas workers with biases will be kept. Raykar and Yu [50] proposed an empirical Bayesian algorithm called SpEM that iteratively eliminates the spammers and estimates the consensus labels based only on good annotators for crowdsourced labeling tasks. Authors in [52] used two previously pilot tested screening questions to filter spammers. People who failed the qualification tasks were disqualified in their research.

Test question sets used in [52, 77, 79, 82] have their limitations. As stated in [60], although test questions are easy to understand, explain and implement, they are also a coarse instrument. The use of test questions may not identify all spammers, and some proper workers might even be rejected just for being unlucky on test questions.

Recognizing unreliable workers through time analysis as presented in [10, 79] is only applicable when the time each crowd worker spent is available. [83] examined how spammers degrade the consensus labels, but without discussing approaches to deal with it. The problem with [81] is that in order to block bad workers or spammers, a worker has to be tested a sufficient number of times, such as $5J^2$ (where J is the number of classes). Consider for example, if the number of classes $J = 5$, then it requires 125 labels from each worker in order to have high confidence to block the bad worker. However, in real world applications, it is not always applicable to obtain the required

number of labels from each worker as indicated in [81].

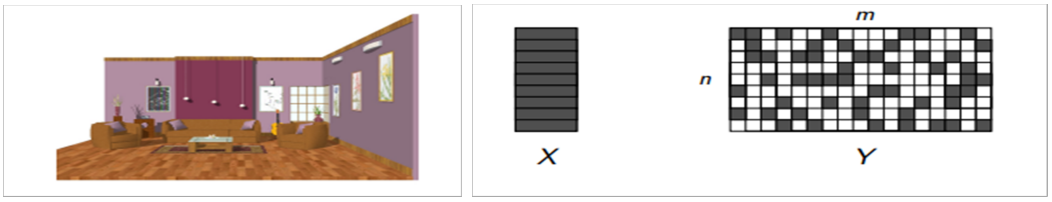
Instead of detecting and removing spammers directly, there has been a lot of research to improve the final consensus results with other indirect methodologies [41, 45, 46, 72, 80]. These researches either block low quality workers by a learning process to estimate the workers reliability, or by making use of repeated labeling. Dawid and Skene [46] proposed to use expectation maximization to estimate the individual error rate in compiling a patient record. In order to infer correct categories for items in data annotation, Carpenter [80] utilized a multilevel Bayesian model in the research. Estimated model parameters include the "true" category of each item, the accuracy in terms of sensitivity and specificity of each annotator, and the difficulty of each item. Sheng et al. [41] examined the improvement in data quality via repeated labeling, and focus on the improvement of training labels for supervised induction. Welinder et al. [72] proposed a generative Bayesian model for annotation process. Each annotator is modeled as a multidimensional entity with variables representing competence, expertise and bias. Whitehill et al. [45] presented a probabilistic model and used it to infer the label of the image, the expertise of each labeler, and the difficulty of each image.

The problem with researches similar to [41] which utilizes repeated labeling is more workers are required for each task, thus the cost increases significantly, especially for large scale crowdsourcing tasks. The other methodologies are mainly targeted at improving consensus results in binary labeling tasks. Although many of them claim it is possible to extend their proposed approaches to categorical data (multiple choices task), the computational complexity will be greatly increased using these frameworks. In addition, these approaches, which are based on probabilistic models, cannot be utilized to labeling tasks with numerical data.

2.2.2 Bias analysis on the crowd workers

Mostly, bias is common when collecting data from crowdsourcing. Bias may be caused by personal preference, systematic misleading, and lack of interest [75]. Many researches have included bias analysis in crowdsourcing models to improve the approximation accuracy [11, 75, 76]. In [75], a Bayesian model, named as Bayesian Bias Mitigation for Crowdsourcing (BBMC), is proposed to capture the sources of bias. BBMC is capable of accounting for labeler bias through preferences and is able to combine data curation and learning into one inferential computation. Gibbs sampling is used as inference approach in the model.

The Bayesian latent feature model, which assigns latent features to labelers to capture their



(a) Task of "Is the guitar behind or next to the couch "

(b) The Structural Illustration of Task Description (X) and Response Matrix (Y) (m is the number of labelers, n is the number of tasks)

Figure 2.3: Example of BBMC model with task "Is the guitar behind or next to the couch"

biases, is specified as: Suppose we solicited labels for n tasks from m labelers. Let task descriptions $x_i \in \Re^d$, $i = 1, ..., n$, be collected in the matrix X. The label responses are recorded in the matrix Y so that $y_{i,l} \in \{-1, 0, +1\}$ denotes the label given to task i by labeler l. As an example, the task could be "is the guitar behind or next to the couch" in Fig. 2.3(a). There could be similar n tasks done through crowdsourcing, thus the X in Fig. 2.3(b) is the task matrix. The responses could be "behind" (-1), "next" $(+1)$, or the labeler didn't give response (0). Y with $n \times m$ dimension shown in Fig. 2.3(b) are responses matrix with rows and columns represent different tasks and labelers, respectively. The highlighted dark cells in each row are responses from various labelers for same task. Labelers express accumulated, shared preferences. Assume latent factor $\gamma_b \in \Re^d$ models effect of preference $b = 1, , K$, and a $m \times K$ binary matrix Z models the factor sharing. If component $z_{l,b}$ in Z equals to 1, it indicates that labeler l expresses preference b. The model account for each labeler l's idiosyncrasies (preference) by assigning a vector $\beta^l \in \Re^d$ to l which accumulates preferences:

$$\beta_l = \sum_{b=1}^{\infty} z_{l,b}\gamma_b \tag{2.5}$$

The likelihood for responses matrix Y then is:

$$p(Y|X, \gamma, Z) = \prod_{(i,l)\in L} p(y_{i,l}|x_i, \gamma, z_l) = \sum_{(i,l)\in L} \Phi(y_{i,l} x_i^T \beta_l) \tag{2.6}$$

where $\Phi(\bullet)$ is the standard normal CDF. The prior distribution of each γ_b is defined to be a zero-mean Gaussian $\gamma_b \sim N(0, \sigma^2 I)$, and Z ($M \times K$ matrix) is governed by an Indian Buffet Process parameterized by α [83], which gives prior:

$$\pi_b|\alpha \sim Beta(\frac{\alpha}{K}, 1), b = 1, ...K \tag{2.7}$$

$$z_{l,b}|\pi_b \sim Bern(\pi_b), l = 1, ...m \tag{2.8}$$

TABLE 2.3

Parameters in TD-HBM model and the definitions

Parameter	Definition
T	Number of consensus tasks
K	Set of possible answers for the task
C	Number of elements in K
F	Dimension of task features
$t_i(\in K)$	True class (label) of task i
$a_i^w(\in K)$	Annotation from worker w for task i
x_i	Task features, automatically generated to represent task characteristics
$\hat{\pi_c^i}$	Likelihood of a worker providing annotations when the true label of the task is c and it has features x_i
Ω_c	F by C matrix of regression coefficients of the logistic regression model
ϵ_c	Mean vector to be added to the multiplication of x_i and Ω_c, a multivariate Gaussian generated with parameter $\mu_c^{epsilon}$ and identity matrix I_C
$\hat{sigma}$	Population score matrix

Gibbs sampling could then be applied to draw samples from posterior distribution of Y, γ, and Z, and finally estimations of response Y and labeler l's preference β^l are obtained.

Authors in [11] introduced and evaluated probabilistic model Task-Dependent Hybrid Bias Model (TD-HBM), which learns a personalized mixture of population and individualistic confusion matrices to represent each worker's bias. The model is described as:

Assume T be the number of consensus tasks and W is a set of worker. K is the set of possible answers for the task and C is the number of elements in K. The TD-HBM model uses the idea of learning a weighting of population and worker confusion matrices to jointly model the relationships among task features, worker characteristics, and bias. The inference problem is predicting the true labels of tasks (t), based on the set of collected labels from crowd (A) and the set of task features (X), the parameters used in building the model and their definition is shown in Table 2.3.

$$Pr(p,t,\hat{\sigma},\hat{\pi},\sigma,\pi,m,\Omega,\epsilon|A,X) = \frac{1}{Z_{th}} \times \prod_{i=1}^{T} \psi_t(p,t_i)\psi_\pi^t(\hat{\sigma_{t_i}^i},\hat{\pi_{t_i}^i},\Omega_{t_i},\epsilon_{t_i}) \times \prod_{w=1}^{W} \psi_\pi(\sigma_{t_i}^w,\pi_{t_i}^w)\psi_a^h(\hat{\pi_{t_i}},\pi_{t_i}^w,a_i^w,m^w) \tag{2.9}$$

where m^w is the mixture weight of worker w, and The potentials $\psi_t, \psi_\pi^t, \psi_a^h$ take the following form:

$$\psi_t(p,t_i) = Dir(p|\gamma)Cat(t_i,p) \tag{2.10}$$

$$\psi_\pi^t(\hat{\sigma_c^i},\hat{\pi_c^i},\Omega_c,\epsilon_c) = N(\epsilon_c|\mu_c^\epsilon,I_C)\prod_{j=1}^{F} N(\Omega_c^j|\mu_c^\Omega,I_C) \times \delta(\hat{\sigma_c^i} - (x_i\Omega_c + \epsilon_c))\delta(\hat{\pi_i^c} - softmax(\hat{\sigma_c^i})) \tag{2.11}$$

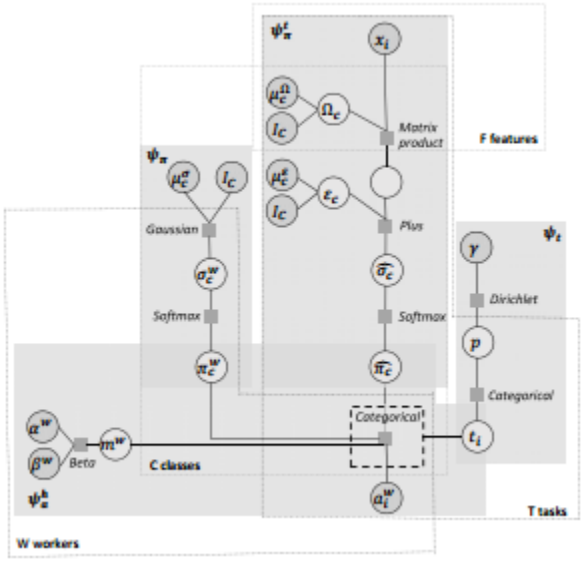


Figure 2.4: Key relationships of TD-HBM model for representing and inferring bias [11]

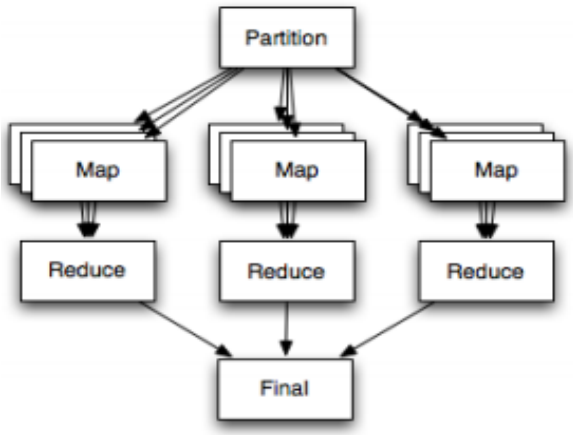
$$\psi_a^h(\hat{\pi_c}, \pi_c^w, a_i^w, m^w) = Beta(m^w|\alpha^w, \beta^w) \times (m^w Cat(a_i^w|\hat{\pi_C}) + (1 - m^w)Cat(a_i^w|\pi_c^w)) \tag{2.12}$$

m^w is generated from a Beta distribution with parameters α^w and β^w. The key relationships of model could be seen in Figure 2.4. The shaded regions mark the potentials defined for inference of posterior distribution. Dashed lines represent plates for workers and tasks. Observed variables are shown with shaded circles and gates are represented with darker dashed squares. For example, the coefficient matrix Ω_c^j is a multivariate Gaussian generated with parameters μ_c^Ω and identity matrix I_C.

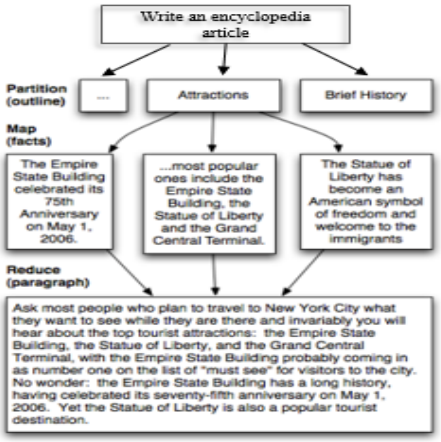
Many other researches [5, 47, 76] proposed bias analysis models, such as in-batch annotation bias discussed by Zhuang and Young [76]. In [76], a factor graph based batch annotation model was proposed to quantitatively capture the in-batch annotation bias, and measure the bias during a crowdsourcing annotation process of inappropriate comments in LinkedIn. In [47], the authors proposed to account for the effects of labeler's bias through a coin flip observation model, which filters a latent label assignment. However, most of these bias models were built based on the binary response applications.

2.3 Related work on labeling complex tasks through crowdsourcing

In general, two main approaches could be utilized in literature to improve the performance of the workers for complex labeling tasks in crowdsourcing systems: a) break down the task into simpler sub-tasks, and b) creating rubrics for workers. Section 2.3.1 gives a review of the work has



(a) Overview of framework for splitting up and recombining complex human computation tasks in MapReduce



(b) Partial Results of Article Writing Task

Figure 2.5: Framework for MapReduce and partial results for article writing task [12]

been done by decomposing tasks into sub-tasks, whereas Section 2.3.2 presented a brief overview of related work in defining expert rubric for the labeling task. In addition, grading process as complex task has been widely discussed in many research, and one of the most popular approaches to deal with grading large scale classes such as MOOCs is peer reviewing. We present some of the work and the trends related to the topic in Section 2.3.3.

2.3.1 Related work on decomposing complex tasks into sub-tasks

Overall speaking, complex tasks are difficult to be done by crowdsourcing through markets like MTurk due to its typical limitations such as low in complexity, independent and requires little time to complete [12]. A lot of research has been conducted to accomplish complex tasks through decomposing them into smaller, independent and less time required sub-tasks [6,12,13,84]. In [12] a framework and a toolkit called CrowdForge were proposed to solve the problem by breaking down the complex task into a sequence of subtasks. Then subtasks are done through MTurk. This approach is a simplified analogous distributed computing systems [85,86] such as MapReduce [87]. Similar to the procedure of MapReduce, as shown in Figure 2.5(a), CrowdForge includes partition, map, and reduce as basic building blocks. In partition step, a high level partitioning of the complex tasks is created. In map step, a specified processing is applied to each item in the partition. For example, a map task for article writing may ask workers to collect facts on a given topic of the article. Finally, reduce task take all the results from the map step and aggregate them into a single result. An example as article writing was given by the authors, and it can be seen in Figure 2.5(b). The task

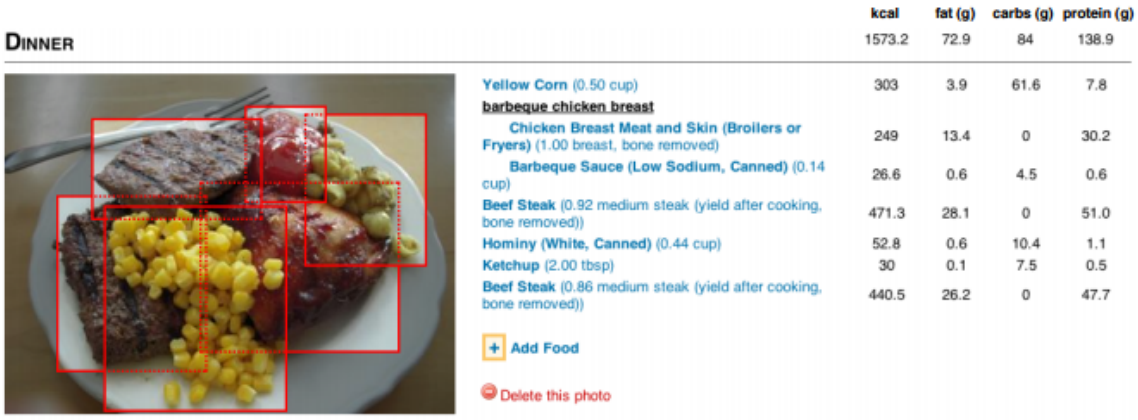
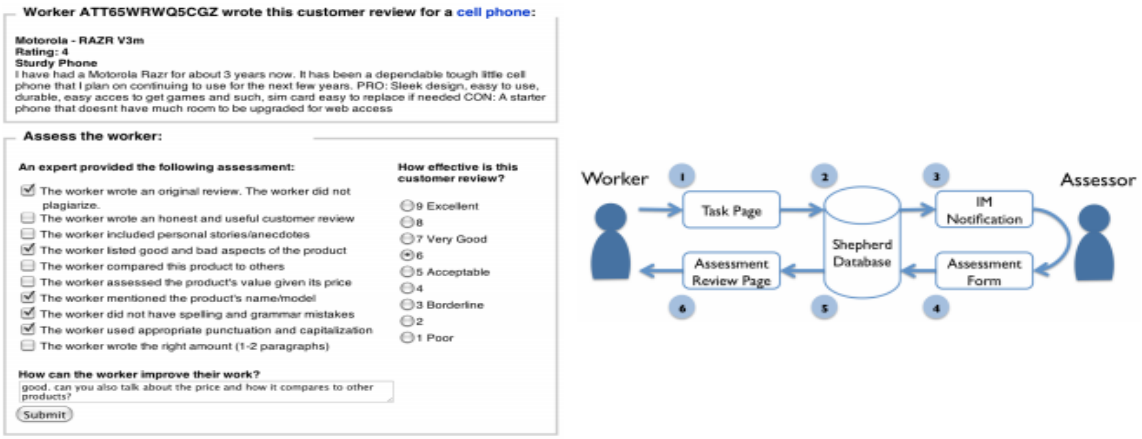


Figure 2.6: The PlateMate user interface. (Users upload their meal photographs, and they are processed through MTurk to get nutrition estimates) [13]

which is writing an encyclopedia, is first partitioned into several sub-tasks, represented as an array of section headings such as "Brief History" and "Attractions". Next, workers are asked to submit a single fact about the section (sub-task). As shown in Figure 2.5(b), the facts under "Attractions" are "answers" for the sub-task which are collected for the crowd. The reduction step will combine the facts collected in map step and create the paragraph for each section.

Authors in [6, 13, 84] presented similar approaches to address the problems encountered while crowdsourcing the solutions to complex tasks. VanHoudnos [6] used crowdsourcing as a novel approach to grade math through Internet, by splitting the expert task into non-expert. In addition, this work also proposed the sub-tasks should be attractive and easy enough for the workers in MTurk. A custom web-application is developed to score student work by organizing decomposed subtasks into an online rating study. The answers from the crowd are then averaged to obtain a single score that is comparable to the teacher assigned score for each submission.

Divide-and-conquer algorithm, which depicts the "decompose, solve and recompose" structure, is proposed in [84] to solve general problems via crowdsourcing. To enable effective and efficient coordination among workers, algorithmic paradigms, such as divide-and-conquer for decomposing a problem into sub-problems, and for composing solutions of sub-problems into solutions, were drawn by the authors. This paradigm is derived from divide-and-conquer algorithm [88, 89], and it enables parallel processing of the problem. In [13], a system called PlateMate is introduced to crowdsource nutritional analysis, such as calories, fat, carbohydrates, and protein, from photographs via MTurk. Users are allowed to upload food photographs on PlateMate. Then three-stage workflow is conducted by the system:



(a) Assessment rubrics used in [14] (b) Shepherd system overview

Figure 2.7: Shepherd system overview and example of rubrics in [14]

(1) Tag takes photos and labels them with boxes drawn around distinct foods on a plate.

(2) Identify matches the tagged box to one or more foods in a commercial nutrition database. Measure returns the nutrition estimates for each identified food.

(3) Finally, after combing these identified parts, users are able to receive the nutrition estimate for the whole food plate. Figure 2.6 shows the user interface given in the work.

Although crowdsourcing complex task by decomposing it into subtasks offers tradeoff between cost and performance, the problem arises when the complex task is un-decomposable. The framework, which decompose the task into sub-tasks and solve them separately by different workers, is no longer available.

2.3.2 Related work on creating rubrics for complex tasks

To the tasks which is not appropriate to decompose into simpler sub-tasks, providing rubrics to workers can be used to improve the work quality in a labeling task. There has been some research done to investigate the impact of offering labeling criteria rubrics to the consensus results [4, 14, 90–92]. Steven et al. [14] suggest in their work that crowds can be shepherded through the use of concrete rubrics. They testified the idea by designing a system called Shepherd which manages workflows for tasks posted to Amazon Mechanical Turk. Figure 2.7(a) gives an example of the assessment rubric to evaluate a product review. The flow chart in figure 2.7(b) shows the overview of the design for Shepherd system. In [4], the authors focuses on studying the effects of rubrics in the domain of design critique. Different from prior work [90–92], which demonstrates the plausibility

TABLE 2.4

List of principle statements that comprise the rubric [4]

Principle Statement	Principle Description
Need to consider audience	The design does not fully consider the target users and the information that could affect their weather-related decisions.
Provide better visual focus	The design lacks a single clear 'point of entry', a visual feature that stands out above all others.
Too much information	Take inventory of the available data and choose to display information that supports the goals of this visual dashboard.
Create a more sensible layout	Information should be placed consistently and organized along a grid to create a sensible layout.
Personalize the dashboard	The design should contain elements that pertain to the particular city, including the name of the city.
Use complementary visuals and text	The design should give viewers an overall visual feel and allow them to learn information from text and graphics.
Needs a clear visual hierarchy	The design should enable a progressive discovery of meaning. There should be layers of importance, where less important information receives less visual prominence.
Thoughtfully choose the typeface and colors	The type and color choices should complement each other and create a consistent theme for the given city.
Other	Freeform critique that does not fit into the other categories.

of obtaining relevant and rapid crowd feedback using rubrics, the work in [4] emphasizes the salient differences between expert and novice feedback providers (crowd workers). The research in [4] used the CrowdCrit system [91] to collect feedback in their experiment. Table 2.4 presented a list of principle statements that comprise the rubric.

Although all the work justified that utilizing rubrics is able to improve the consensus results and work quality of the crowd, they failed to take into account the workers′ reliability during the process. Due to the unpredictability feature of the workers on crowdsourcing platforms, it is still necessary to consider the factor of trustworthiness of them.

2.3.3 Peer reviewing for complex tasks

To relieve the grading burden of the experts in traditional way [93–95], especially with the popularity of open online classes such as MOOCs, peer reviewing has been proposed as a new way to addressing the grading tasks [5,7,15,16,96,97]. Peer grading or peer reviewing [98,99] is an approach in which students also work as graders in grading a small number of submissions completed by other students according to the benchmarks provided by the instructor [16]. As stated in [96], "students are trained to be competent reviewers and are then given the possibility of providing their classmates with personalized feedback on expository writing assignments". Most of the peer evaluation systems proposed nowadays are similar to this concept.

Piech et al. [5] developed three statistical models, PG1 (grader bias and reliability), PG2 (temporal coherence) and PG3 (coupled grader score and reliability), for estimating true grades, grader biases and reliabilities in peer grading. The models put prior distributions over latent variables

such as true score, grader's reliability and bias. They are shown as follows [5]:

Model PG1:

$$\text{(Reliability) } \tau_v \sim G(\alpha_0, \beta_0) \text{ for every grader } v$$
$$\text{(Bias) } b_v \sim N(0, 1/\eta_0) \text{ for every grader } v$$
$$\text{(True score) } s_u \sim N(\mu_0, 1/\gamma_0) \text{ for every user } u$$
$$\text{(Observed score) } z_u^v \sim N(s_u + b, 1/\tau_v) \text{ for every observed peer grader}$$

where G referes to a gamma distribution, and α_0, β_0, and γ_0 are hyper-parameters for the priors. The PG1 model forms the relation between the parameters such as grader reliability, bias and true score. For example, if the grader's reliability and bias are known, it is possible to infer the true score, based on the observed score.

Model PG2:

$$\tau_v^{(T)} \sim G(\alpha_0, \beta_0) \text{ for every grader } v$$
$$b_v^{(T)} \sim N(b_v^{(T-1)}, 1/\omega_0) \text{ for every grader } v$$
$$s_u^{(T)} \sim N(\mu_0, 1/\gamma_0) \text{ for every user } u$$
$$z_u^{v,(T)} \sim N(s_u^{(T)} + b_v^{(T)}, 1/\tau_v^{(T)}) \text{ for every observed peer grader}$$

Correlation of graders$'$ bias between two consecutive assignments has been examined and a Pearson correlation of 0.33 was found by Piech et al. Compared to PG1, model PG2 takes into account the relationship of a grader's bias and reliability at homework T to that at homework $T^{'}$. That means, PG2 builds the model in the time domain, whereas PG1 is in the batch mode.

Model PG3:

$$b_v \sim N(0, 1/\eta_0) \text{ for every grader}$$
$$s_u \sim N(\mu_0, 1/\gamma_0) \text{ for every user}$$
$$z_u^v \sim N(s_u + b, \frac{1}{\theta_1 s_v + \theta_0}) \text{ for every observed peer grader}$$

PG3 explores the relationship between one's grade and one's grading ability, which depend on her own grade. It extends PG1 by introducing new dependencies, which enables estimation of grader's ability by her submission score. After building the models based on the prior distributions, Gibbs sampling [73] was used to produce samples from desired posterior distribution, which are then able to be utilized to estimate true scores of the submissions. For instance, given samples $s_u^1, s_u^2, ..., s_u^T$ from the posterior distribution over the true score of submission u, the estimated true score may be: $\hat{s}_u = \frac{1}{T}\sum_{t=1}^{T} s_u^t$.

A system called CrowdGrader is developed in [7] to explore the use of peer feedback in grading assignments. It introduced a Vancouver algorithm (derive from [38]) which relies on a reputation system to aggregate the peer reviewing grades. This algorithm is based on the fact which is known as minimum variance estimator: Suppose there are uncorrelated estimates $\hat{X}_1, ..., \hat{X}_n$ of a quantity of interest, where $\hat{X}_i$ is a random variable with average x and variance v_i. It is possible to obtain an estimate of x that has minimum variance by averaging $\hat{X}_1, ..., \hat{X}_n$ while giving each $\hat{X}_i$ weight as $1/v_i$, for $1 \leq i \leq n$. That means, the minimum variance estimator $\hat{X}$ of x is formulated as:

$$\hat{X} = \frac{\sum_{i=1}^{n} \hat{X}_i/v_i}{\sum_{i=1}^{n} 1/v_i} \tag{2.13}$$

where the variance of the estimator is:

$$var(\hat{X}) = (\sum_{i=1}^{n} \frac{1}{v_i})^{-1} \tag{2.14}$$

This estimator suggests that it is possible to weigh the input provided by student i in proportion to $1/v_i$. The fact suggests that it is possible to weigh the input provided by grader i in proportion to $1/v_i$ while aggregating the observed grades. For instance, if grader i and $(i+1)'$s variances are measured as 0.7 and 0.5, respectively, and the grades they provided for submission j are 80 and 95. The aggregating process to get final consensus grade is then: $80 \times (\frac{0.7}{0.7+0.5}) + 95 \times (\frac{0.5}{0.7+0.5}) \approx 86$.

Ashley and Goldin [15] applied Bayesian data analysis to model a computer supported peer review (CSPR) process in a legal class. In this work, the authors proposed peer review as social network, in which students focus on the topic of review criteria, and it may be seen as a directed graph. Figure 2.8 shows an example of the graph. Each node is a student, and an arc connects a reviewer to an author. Feedback given by a reviewer is called outbound feedback, and feedback received by an author is said to be an inbound feedback. Arcs with solid lines represent the review relation, and the dotted line arcs means the rate provided by the author to the reviews. Three reviewers with the inbound arc labeled with "i" reviewed student 22's work. The outbound arc labeled with "o" represents student 22 reviewed other student's work. Similarly, the inbound arc labeled with "i b-r" is the rate from other student to student 22's review, and outbound "o b-r" arc is the rate given by student 22 to the reviewers who reviewed his work. In order to extract useful information from the peer review data on a midterm exam in the Intellectual Property (IP) class, hierarchical Bayesian modeling [100] is applied to build models in order to estimate parameters such as quality of student works and the variability across reviewers. Each model maps between the

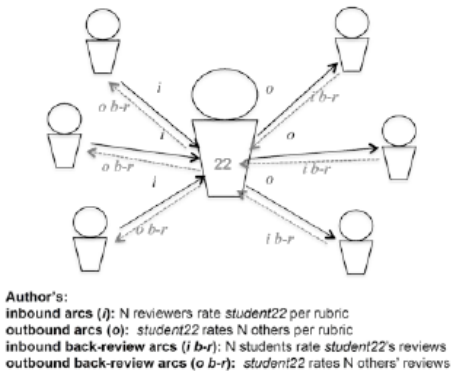


Figure 2.8: Peer review as social network [15]

instructor's independently assigned exam scores and the peer ratings. MCMC sampling computes all the quantities of interest at once using all the available data.

In [96], a web-based peer review program called Calibrated Review (CPR) system is introduced to improve student learning. By introducing the students to the peer review system, the CPR program, which enables instructors to incorporate writing assignments into their courses regardless of the class size, allows these students to have more opportunities to gain access to frequent writing and critical thinking exercise, insight into professional practices, and personalized learning support, etc. By adopting "double blind" review process and ensuring student feedback is consistent and reliable, the CPR system reduces the faculty workload greatly.

Mi and Yeung [16] studied both cardinal and ordinal aspects of peer grading within one framework. Cardinal grading is giving absolute judgement, such as provides grade 80 for a submission is under cardinal setting. Whereas ordinal grading is relative judgement, for example, instead of giving grades for the students, offering judgement as student i's submission is better than student j is ordinal grading. The authors proposed novel extensions to existing models for cardinal peer grading. These extensions not only provide superior performance for cardinal peer grading, but also outperform conventional ordinal evaluation models. The approach combined cardinal and ordinals models by augmenting ordinal models with cardinal predictions as prior.

Cardinal peer grading used models [16] which are extensions of the probabilistic models proposed by Piech et al [5], as we described earlier. The key difference between these models with models in [5] is the way in which the relationship between the reliability and the true score of a grader is modeled. In [5], the reliability of every grader is fixed to be the same value. However, in [16], a probabilistic relationship is imposed between the reliability and true score of a grader. The models proposed in [16] named as PG4 and PG5 are denoted as follows:

Model PG4:

$$\text{(Reliability) } \tau_v \sim G(s_v, \beta_0)$$
$$\text{(Bias) } b_v \sim N(0, 1/\eta_0)$$
$$\text{(True score) } s_u \sim N(\mu_0, 1/\gamma_0)$$
$$\text{(Observed score) } z_u^v \sim N(s_u + b, 1/\tau_v)$$

Model PG5:

$$\tau_v \sim G(s_v, 1/\beta_0)$$
$$b_v \sim N(0, 1/\eta_0)$$
$$s_u \sim N(\mu_0, 1/\gamma_0)$$
$$z_u^v \sim N(s_u + b, 1/\tau_v)$$

In PG4, a grader's reliability follows the gamma distribution with rate parameter β_0 and the true score s_v as the shape parameter. Meanwhile, PG5 assumes reliability follows the Gaussian distribution with s_v as mean and $1/\beta_0$ as variance. The same graphical model are given by the authors for these two models, and it is presented in Figure 2.9. The plate notation is used to represent separately the gradee ("U") and grader ("V") for clarity. The reliability, bias and true score are latent variables in the graphical model, and observed score z_u^v is the only observed variable. The hyper-parameters $\mu_0, \gamma 0, \beta_0, \eta 0$ are used for specifying the probability distribution of latent variables. The arc represents the relationship between parameters and variable. For example, parameters μ_0 and γ_0 contribute to the gradee's true score s_u in the model, thus there are arcs originating from μ_0 and γ_0, and points at s_u. Similarly, s_u, b_v and τ_v are parameters deciding observed score z_u^v, and the relation could be seen as presented with arcs in Figure 2.9. Finally, while combining cardinal and ordinal models, Bradley-Terry model [101, 102] was utilized.

Peer reviewing experiments have shown promise, but it may also cause problems since same group of students involved as both gradees (whose submissions need to be graded) and graders in the grading process. As what Richard Smith proposed in his paper, "the practice of peer review is based on faith in its effects, rather on facts" [103]. Suppose all the students in one class try to give good grades to each other, then with the absence of the true grades provided by the instructor or teaching assistant used for compare, the consensus grades would be the submissions′ grades. However, these grades actually cannot give correct estimation of the true grades. To overcome the defect of peer reviewing, we introduce crowdsourcing among experienced domain graders as the grading approach for large scale classes.

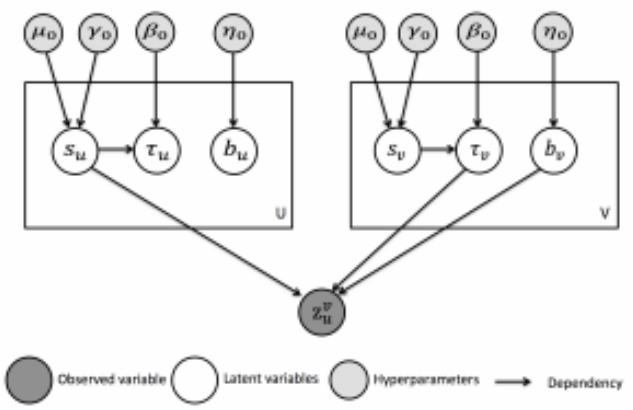


Figure 2.9: Graphical model for PG4 and PG5 [16]

2.4 Data sets used in crowdsourcing research

In general, there are two categories of the data sets utilized for labeling tasks through crowdsourcing: 1) simulated (synthetic) data set and b) real world data set. We first show some of the related works in generating simulated crowdsourcing data in Section 2.4.1. Real world datasets used in crowdsourcing research are then presented in Section 2.4.2.

2.4.1 Synthetic data sets

The goal of using simulated data set is to study the accuracy of mathematical approximations and the effect of assumptions being violated. The simulation process use methods based on random numbers to simulate a real-world process of interest on the computer[2]. Alfaro and Shavlovsky [7] evaluated their crowdsourcing algorithm for grading tasks on a simulated data set, which consists of 50 graders and 50 submissions with each user reviewing 6 submissions. They simulated the true quality q_i of each submission i from a normal distribution. Each user was assumed to have a character variance v_j and the grade q_{ij} assigned by grader j to submission i be equal to $q_i + \Delta_{ij}$. Δ_{ij} is the deviation from true grade which is simulated based on v_j.

Chen and Bennett [104] conducted experiments with simulated data to demonstrate the proposed active learning strategy for pairwise ranking aggregation. They created 100 objects, each with an underlying true score in the range of 1 to 100. The authors also created 100 annotators, whose true qualities follow a beta distribution. 400 pairs of objects are then randomly sampled for comparing, with each pair compared by 10 annotators. Thus, they gathered 4000 compared pairs for analyzing by developed models.

[2]http://www.esg.montana.edu/R/rsim.pdf

TABLE 2.5

Public datasets used in crowdsourcing research

Dataset		Categories	Items	Workers	Labels
Image Labeling Datasets	ESP	-	100,000	-	-
	ESP Lite	-	9376	-	-
	WVSCM	2	159	17	1221
	WB	2	108	39	4212
	Face Sentiment	4	584	27	5256
NLP Datasets	AC2	4	333	269	3317
	BM	2	1000	83	5000
	RTE	2	800	164	8000
	TEMP	2	462	76	4620
	WSD	3	177	34	1770
Social Tagging Datasets	MovieLens	5	10681	71567	10 million
	TagATune	-	100	54	-
Other Datasets	SpamCF	2	100	150	2297
	HC	3	3275	722	18479
	HCB	2	3275	722	18479

Tang and Lease [44] simulated a set of crowd workers to evaluate the effectiveness of their proposed semi-supervised Nave Bayes framework, which is used to obtain the high quality consensus labels. They generated a synthetic data set for binary classification tasks with 8000 items/examples uniformly assigned to each class. A pool of 800 worker is generated, each with a simple Bernoulli accuracy parameter $p_k \sim U[0.3, 0.7]$. The number of labels per example is randomly set between 2 to 8.

2.4.2 Real world data sets

Real world data set is usually unpredictable in crowdsourcing applications, which is different from simulated data set. This is because characteristics of the model developed to obtain simulated data is already known by the researchers, thus the simulation results are predictable. However, analyzing real world data provides valuable insights into the usage and comparative effectiveness of the developed algorithms and methodologies.

We begin by identifying and describing a number of public datasets that are available online for related crowdsourcing research nowadays. We categorize the datasets based on their task goals, and they are presented as follows. Table 2.5 provides summary statistics for each dataset.

Image Labeling Datasets. Von Ahn et al. contributed a list of 100,000 images with English labels from their **ESP**[3] Game [105]. The ESP game is a two-player image labeling game. Both players are asked to assign labels to the same image without the benefit of communication, and only labels

[3]http://www.cs.utexas.edu/~grauman/courses/spring2008/datasets.htm

TABLE 2.6

Description of data sets collected in Human Computer Interaction (HCI) classes on Coursera by Piech et al [5]

	First HCI	Second HCI
Students	3,607	3,633
Assignments	5	5
Submissions	6,702	7,270
Peer Grades	31,067	32,132

they assign common is accepted. Chen et al. developed the **ESP Lite**[4] game, which is similar to the ESP game, and collected statistics for players playing the game [106]. **WVSCM**[5] is a dataset which includes AMT binary judgements distinguishing whether or not face images smile [45]. **WB**[6] dataset has AMT binary judgements indicating whether or not a water-bird image shows a duck [72]. **Face Sentiment**[7] gives the data set collected for the task of identifying the sentiment for a given face image. The sentiment is about whether it is neutral, happy, sad or angry [106].

NLP Datasets. Ipeirotis et al. contributed a Natural Language Processing (NLP) dataset called **AC2**[8] with AMT judgements for website (ordinal) ratings G, PG, R, X, B [31]. The task is providing labels for the presence of adult content on the web pages. **BM**[7] contains negative/positive (binary) sentiment labels 0, 1 assigned by AMT workers to tweets [106]. **RTE**[9], **TEMP**[9], and **WSD**[9] provide AMT labels [42]. RTE includes binary judgements for textual entailment, consider for example, whether one statement implies another. This task replicates the recognizing textual entailment task originally proposed in [107]. TEMP includes binary judgements for temporal ordering such as whether one event follows another. WSD includes ternary multiple choice judgements for selecting the right sense of a word given an example usage. The data set is derived by annotating part of the SemEval Word Sense Disambiguation Lexical Sample task [108].

Social Tagging Datasets. **MovieLens**[10] dataset holds 10 million ratings and 100,000 tages for 10681 movies by 71567 users. It is released by GroupLens research group in the Department of Computer Science and Engineering at the University of Minnesota. Law et al. released the research dataset for a music and sound annotation game called **TagATune**[11] [109]. It is a two-player online

[4] http://hcomp.iis.sinica.edu.tw/dataset/dataset_esplite20100101.php
[5] http://mplab.ucsd.edu/∼jake/DuchenneExperiment/DuchenneExperiment.html
[6] http://www.vision.caltech.edu/visipedia/CUB-200.html
[7] http://people.csail.mit.edu/barzan/datasets/crowdsourcing/
[8] https://github.com/ipeirotis/Get-Another-Label/tree/master/data
[9] https://sites.google.com/site/nlpannotations/
[10] https://grouplens.org/datasets/movielens/
[11] http://tagatune.org/

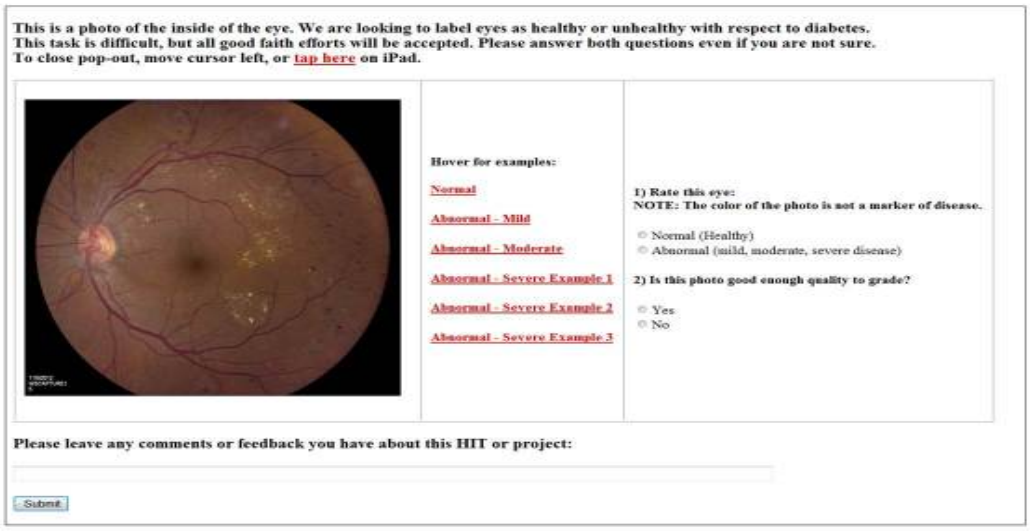


Figure 2.10: Example of Fundus photo posted on MTurk for classification in [17]

game which collects data about sound clip and music, such as tags and similarity comparisons. Also, Körner and Strohmaier [110] presented a list of social tagging datasets which are available for research.

Other Datasets. **SpamCF**[8] includes binary AMT judgements about whether or not an AMT HIT should be considered as a "spam" task, according to their criteria [111]. **HC**[12] dataset has AMT ordinal graded relevance judgements for pairs of search queries and web pages: not relevant, relevant, and highly-relevant [44, 112]. Whereas **HCB**[13] conflates relevant classes to produce only binary labels [113].

Although most of the data sets for complex tasks are not publicly available, we would still like to provide an overview about them. Piech et al. [5] gathered the data sets from two consecutive Coursera offerings of Human Computer Interaction (HCI) for the peer grading system. The description of the data sets are shown in Table 2.6. In each class, 5 assignments are collected from students. Each student evaluated 5 randomly selected submissions, the amount of grades received is presented in the "Peer Grades" row of the table.

In order to test the proposed model for fundus photograph labeling, Brady et al. [17] developed an interface for fundus photo classification on MTurk to collect data. Figure 2.10 gives an example of the fundus photo posted on Amazon MTurk. It asked the workers to evaluate the photos according to the instructions set. There were a total of 230 labeling items evaluated by workers on

[12]https://docs.google.com/document/d/1J9H7UIqTGzTO3mArkOYaTaQPibqOTYb_LwpCpu2qFCU/edit
[13]http://ir.ischool.utexas.edu/square/data.html

TABLE 2.7

Data set gathered by crowdsourcing the math grading task on MTurk [6]

Grade	Students	Questions	No. HITs/Question	Size	No. Scores
3	38	2	1	5 30	380 2,280
6	49	2	1	5 30	490 2,940
8	43	2	2	5 30	860 5,160

MTurk in three phases. Phase I is for the purpose of filtering of graders, in which workers who successfully completed 100 prior tasks were selected. In Phase II, 500 unique evaluations are received, and in Phase III, fifty draws of 1-50 workers were obtained.

To verify the concept of grading math through crowdsourcing, VanHoudnos [6] conducted the data-collection experiment on Amazon MTurk. Submissions from the students of 3^{rd}-8^{th} grade math classes of a local school are collected for grading. The description of the data set is summarized in Table 2.7. Students$'$ answers (submissions) for 2 questions are obtained from different Grade. The questions from Grade 8 were split into two HITs (two grading tasks), and then all submissions are posted on MTurk for grading.

In addition to the simulated data set, Alfaro and Shavlovsky [7] also used the real world data set to evaluate the proposed Vancouver algorithm, which is used for aggregating the collected grades into consensus grades. The data set consisted in 5 homework assignments from an Android class (CMPS 121); 5 assignments from C++ class (CMPS 109), and one assignment from Java Class. The number of submissions and reviews are presented in Table 2.8. The "ReqRevs" column in the table represents required reviews for each submission; "MinRevs" represents the minimum reviews received by a submission, and "AvgRevs" is the average number of reviews got by each submission.

Yuan et al. [4] recruited 15 students from a design course to each create a weather UI dashboard. Later, 36 crowd workers including 12 from Upwork [114] and 24 from MTruk are recruited to provide feedback for the designs submitted by students. Each Upwork worker was requested to critique 8 designs, and every MTurk worker was asked to critique 4 designs. Dow et al. [14] recruited 207 MTurk workers to write consumer reviews for 6 products they own and all the reviews were reposted to MTurk for a crowd assessment. Up to 5 workers judged each review using expert rubrics. There were total 408 crowd judges and 2229 valid assessments for final analysis. The authors in [54] developed word processing interface called Soylent which aids the writing process by integrating

TABLE 2.8

Data set collected from grading of coding assignments in [7]

Assignment	No. Submissions	ReqRevs	MinRevs	AvgRevs
CMPS 121 hw1	60	6	2	5.4
hw2	61	6	2	5.3
hw3	68	6	0	4.8
hw4	62	6	6	6.1
hw5	57	6	5	5.3
CMPS 109 hw1	102	5	0	4.6
hw2	97	5	3	4.6
hw3	91	5	4	5.1
hw4	97	5	3	4.6
hw5	90	5	4	5.1

crowd workers from MTruk into Microsoft Word. They interacted with 8809 Turkers across 2256 different tasks for the project experiments.

2.5 Towards learning true labels from noisy crowdsourced data

From the work discussed for crowdsourcing systems, it could be seen that noise is inevitable due to the diverse skill levels and motivations of the workers. To tackle the low quality problem of the collected crowdsourced data, a vast majority of works have been proposed from repeated labeling to worker filtering in order to improve the performance of the consensus results. Almost all the methodologies are trying to do two things: a) Estimating worker reliability, and b) Inferring the true labels by utilizing the estimated worker reliability. It is desirable to use labels received from reliable workers to infer truths for items. Worker filtering techniques are developed to deal with the unreliable workers so that only reliable workers can be maintained for further research.

Worker filtering methods in literature can be divide into two categories: 1) Detecting and removing spam workers, and 2) Recognizing biased workers. Most of the research is carried out in either one of them, and for binary labeling tasks. It is possible to have a methodology to integrate spam removing and bias detection to give further improvement of the final results. Also, there a need to have a framework which could infer true labels from both numerical and categorical crowdsourced data.

For complex labeling tasks, providing expert rubrics is an effective way to improve the work performance of the crowd workers. However, the existing works does not investigate the worker reliabilities in the process. Thus, it is necessary to build a system which takes into account this factor in truth inferring process.

In addition, it is very likely to have a small set of gold labels pre-known in real world crowdsourcing applications. We can benefit from a semi-supervised learning approach for worker selection by utilizing this subset of gold truths, and it then further improves the quality of consensus labels.

CHAPTER 3

IMPROVING PERFORMANCE OF LABELING PROCESS – AN INTEGRATING APPROACH WITH SPAM REMOVING AND DEBIASING

The crowd workers with unpredictable behaviors, including variety of spam and biased workers, might greatly degrade the quality of consensus labels. This chapter proposes a crowdsourcing framework based on spam detection and de-biasing algorithms, to improve the accuracy of the consensus results. Furthermore, optimization of the number of workers while maintaining the quality of labels is also presented.

3.1 Introduction

Crowdsourcing attracts different types of workers, including spam workers (also called spammers). There has been some research reported the worker spamming questions [10, 50, 77, 82]. Spam workers are labelers with poor work performance, which gives low quality answers for tasks. As an example, a worker who frequently gives random labels for items can be regarded as a spam worker. There are various reasons for the existence of spam workers: workers do not have enough knowledge to understand the tasks or criteria of completing the tasks; do not want to spend efforts and time to finish the job; or they are just malicious workers [50]. The spamming problem proposes negative influences and new challenges on crowdsourcing platforms. As discussed in Chapter 1, redundancy is one of the commonly used approaches to deal with the problem. That means, the same task is completed by multiple workers. Consider for example, an image labeling task requested labels for a set of images about whether each of them is appropriate for children. The possible label set is defined as 1 (No adult content), 2 (Content requires parental guidance), 3 (Mainly for adults). Figure 3.1 gives an overview of the repeated labeling process for this image labeling task. First, the task is posted on the crowdsourcing platform to request labels for each of the image. Multiple labels are collected for each item, and these labels will then be aggregated to get one single consensus label for this image. However, the more workers are used for crowdsourcing, the more cost there will be. Another method is the usage of predefined "gold" data set [77, 82]. In this approach, a standard

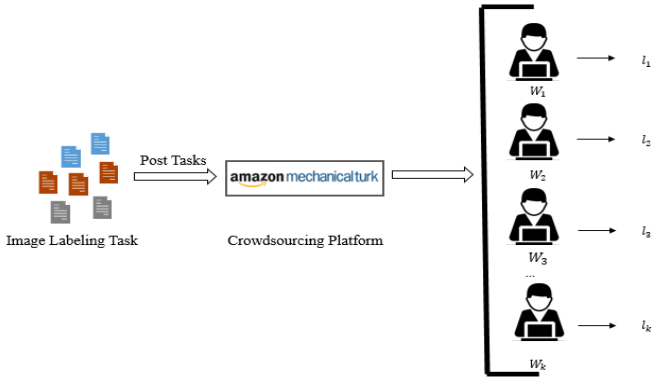


Figure 3.1: Repeated labeling for image labeling through crowdsourcing

data set as training data has been predefined by the experts, with ground truth answers to the data known. If the number of answers, which provided by a crowd worker, for the questions in training data exceeds the threshold set by experts, the worker is detected as spammer and are rejected. By using "gold" set, it is possible to recognize some of the spammers, and thus might be able to reduce the number of workers for crowdsourcing tasks compare to redundancy approach. Although possible, the detection of spam workers by gold set is not guaranteed [60].

Unlike spam workers, which significantly degrade the accuracy of the crowdsourcing results, and their voted labels cannot be used, biased workers' labels are valuable and recoverable. Biased workers usually offer labels with biases, but it is possible to extract the useful information from their answers. Bias may be caused by personal preference, systematic misleading, or lack of interest [10]. For a skewed dataset, spam workers are possible to give low error rates by giving all labels into the majority class. For example, if the true label of most items are '3', then those spam workers who give '3' to all the items will be misclassified as high quality workers. Since in this case, the spam workers do have low error rates. However, biased workers might show higher error rates when comparing to the spam workers in this case. As an example, consider two workers with w_1 as biased and w_2 as spam worker. Let a set of items which need to be labeled with true labels as $[1, 2, 4, 4, 4, 4, 3, 4, 5, 5]$ on the scale from 1 to 5. Assume the labels offered by w_1 are $[4, 4, 4, 4, 4, 4, 4, 4, 4, 4]$, and w_2 provided labels as $[2, 3, 5, 5, 5, 4, 4, 5, 5, 5]$. It can be seen that w_1 has accuracy as 0.5, and w_2 as 0.3. However, we could obtain accuracy as 0.7 if we correct the labels from w_2 via increasing his labels by 1 level up.

In this chapter, we targeted at tasks with numerical labels for the items. A novelty bias recognition approach is developed to iteratively de-bias the labels from biased workers. In addition, a spam detection algorithm is adopted and tuned from existing literature to better recognize and

remove spam workers here. By mapping the continuous rating scale into discrete ones, we also proved the extensibility of the approach from discrete labeling tasks into continuous scoring problems. We then present an integrated approach, which is based on spam removing and bias detection algorithms, to give significant improvement on the quality of the consensus labels for the items. Lastly, we optimize the number of workers for labeling tasks, while maintaining the quality of the final consensus labels. As a result, the cost for crowdsourcing is able to be reduced with fewer workers used for labeling.

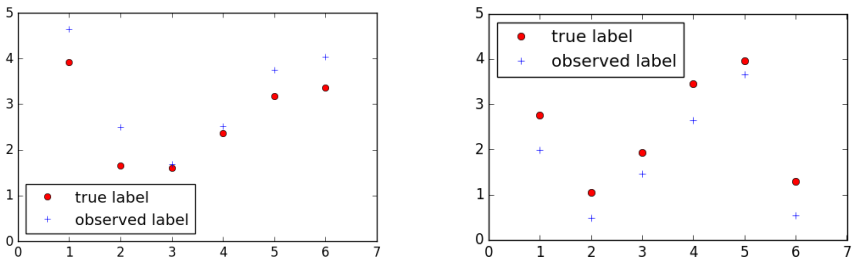
The rest of the chapter is organized as follows: Section 3.2 presents the bias detection algorithm in order to correct the labels offered by biased workers. Section 3.3 presents the spam removing method which can be utilized to filter different types of spam workers among the crowd. Section 3.4 shows the way of how to optimize the number of workers. Section 3.5 gives an introduction of aggregating algorithms used to obtain consensus labels, and presents the proposed Spam Removing and De-biasing Framework (SRDF) to improve the quality of estimated labels. Experimental evaluation and results are presented in Section 3.6. Section 3.7 presents the chapter summary.

3.2 Bias detection algorithm

Bias is defined as any tendency which prevents unprejudiced consideration of a question [115], and it is common in the research process. Bias may be caused by personal preference, systematic misleading, and lack of interest [75] in the crowdsourcing process. Directly using biased data would greatly degrade the accuracy of the final results. However, the labels provided by biased workers while crowdsourcing labeling tasks could be recoverable and valuable. It is possible to extract the useful information – estimation of true label of the item from the biased answers. In order to de-bias the labels collected from biased workers, it is desirable to first detect the bias behavior of the labelers. For instance, assume the true labels for five different items should be $\{5, 4, 4, 4, 4\}$. The biased worker w provide labels as $\{4, 3, 3, 3, 3\}$. If we de-bias the labels provided by w through increasing one level to all answers from him, then the error rate becomes from $^5/_5 = 1$ to $^0/_5 = 0$, and accuracy is 100%, which is defined as the percentage of labels that are correctly given.

Consistent and similar behavioral errors are evaluated as bias patterns, and it may be identified by comparing the labels obtained from workers with true labels. Bias is then reduced from the labels given by the biased workers, whom with detected bias patterns.

The bias pattern is defined in our research as: for all the labels provided by worker w, if of his/her labels for item i is consistently higher (or consistently lower) than the true label of the item,



(a) Positive bias pattern (b) Negative bias pattern

Figure 3.2: Bias patterns

then we say the worker w has a bias pattern. There are two different types of bias pattern:

(i) Most of the worker's answers are higher than the true labels, we call this pattern as "positive bias pattern";

(ii) Worker's answers are lower than the true labels, which we call "negative bias pattern".

Figure 3.2 gives an example for each of these bias patterns. From the definition of bias pattern, it could be seen that unbiased worker does not mean that the worker gives labels exact same as true labels. Instead, unbiased means they do not have fixed behavior pattern on the bias errors they make. Figure 3.3 gives an example of comparing biased and unbiased worker. In Figure 3.3, the small circles represent true labels, whereas the "+" sign is observed labels. Worker in figure 3.3(a) has a positive bias pattern, whereas 3.3(b) has no bias pattern presented.

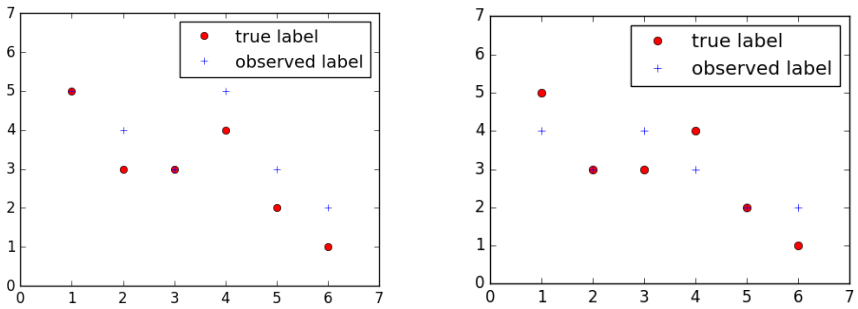
We propose the pattern detection approach to detect and measure bias pattern as the percentage of label values that lies within two standard deviations around the μ, which is differences between the true labels and observed labels. Below we formalize the way to recognize the bias pattern:

Suppose there are n different items labeled by worker w, for item i. True label of item i is given as $t_i \in T$, where T is the true labels set. O is the set of observed labels from w, where $o_i \in O$ is the label given by w to i_{th} item. Denote $diff(t_i, o_i)$ as the difference for each pair of ti and o_i. Calculate:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} diff(t_i, o_i) \tag{3.1}$$

$$\sigma^2 = E((O-\mu)^2) \tag{3.2}$$

$$(\mu - 2\sigma, \mu + 2\sigma) \tag{3.3}$$



(a) Example of biased worker

(b) Example of unbiased worker

Figure 3.3: Comparison of biased and unbiased workers

where μ is the mean of the differences between observed labels from workers and true labels for item. Interval (3.3) represents the width around the mean of the differences of the true labels and observed labels within standard deviations. In statistics, if $\mu - 2\sigma$ calculated from equation (3.1) and (3.2) is greater than 0, then we could say more than 95% of the difference values ($diff(t_i, o_i)$) are positive. In other words, more than 95% of the worker's labels are higher than the true labels. We state that this worker has a positive bias pattern with 95% confidence. Similarly, if the result of $\mu + 2\sigma$ is negative, we have 95% confidence that the worker has a negative bias pattern. In real application, the interval should be adjusted through heuristic experiments with best results, such as tune the interval into $(\mu - \sigma, \mu + \sigma)$. Figure 3.4 gives an illustration of the pattern detection approach (take 95 percentage as example). The normal distributed curve is calculated $diff(t_i, o_i)$'s for each worker. The red dotted line denotes the positive bias pattern condition, whereas the green dotted line represents the negative bias pattern requirement.

Observed labels from the workers with negative or positive bias patterns should be de-biased to give a better performance before applying aggregating algorithm. The approach of correcting labels for biased graders varies on different data sets. The approaches for de-biasing might be subtracting: a) $min(diff(t_i, o_i))$; b) $max(diff(t_i, o_i))$; c) median of $(diff(t_i, o_i))$ or d) average of $(diff(t_i, o_i))$ from the observed labels. The most appropriate method for given application would be determined through our experiments and formalized as experimental heuristics. In our research, the median of $diff(t_i, o_i)$, with $t_i \in T$, $o_i \in O$, is worked as bias to be subtracted from observed labels to obtain the corrected/de-biased labels. The corrected labels are utilized to re-estimate the consensus labels, and the newly computed estimated labels are again used for the de-biasing approach. The iterations stop when the results converge.

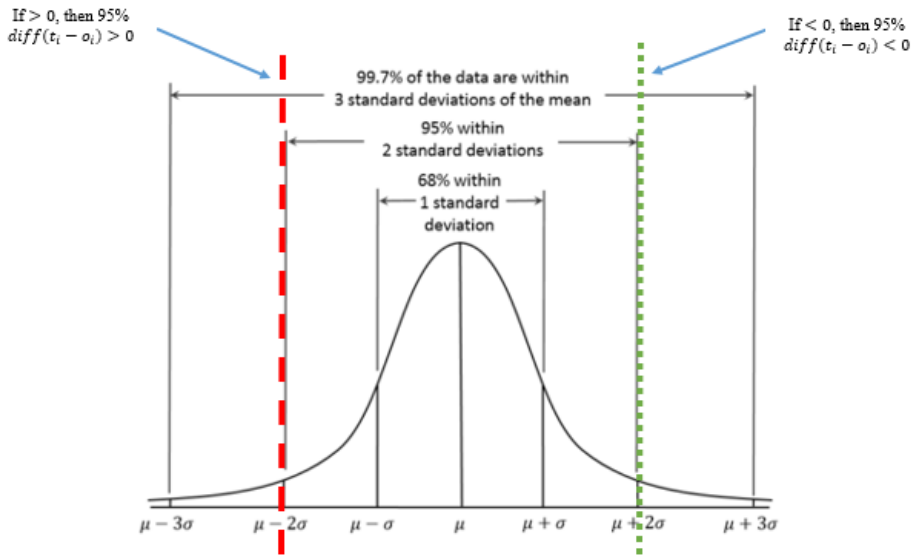


Figure 3.4: Illustration of pattern detection approach: Red dotted line represents the case of positive bias pattern, and green dotted line represents the case of negative bias pattern

3.3 Spam filtering

Before giving details about the spam removing approach, we will first show the categorization of the crowd workers based on their behavior patterns. Afterwards, spam detection algorithm proposed by Vuurens et al. [60] will then by tuned and adopted to recognize different types of spam workers.

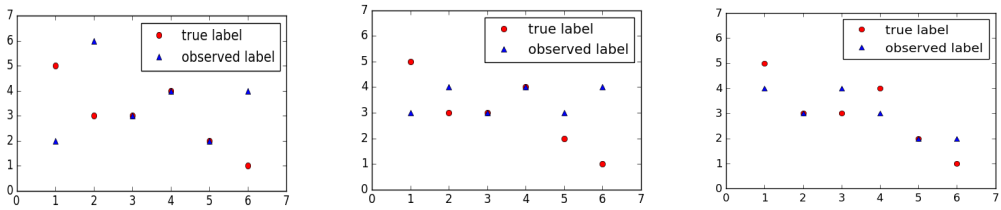
3.3.1 Worker categorization

Similar to [60] and [78], workers are classified into several different categories with respect to spamming problem:

(1) *Random Spammer*: gives random labels for the items. This type of worker will change the labels they give for items frequently, and they don't have any behavior pattern.

(2) *Uniform Spammer*: provides same labels for most of the items. Uniform spammers have a repeating labeling behavior pattern. For example, workers who always give B to most of the items, or repeating "BC" label pattern are uniform spammers. Except for the case that true labels follows this pattern, such as true labels are: B, C, B, C, B, C, then labels with uniform "BC" pattern are no longer spammers.

(3) *Sloppy Worker*: this type of worker offers low accuracy labels. However, they neither give totally



(a) Random spammer (b) Uniform spammer (c) Sloppy worker

Figure 3.5: Examples of random spammer, uniform spammer, and sloppy worker

random labels, nor have repeating label pattern. They might be malicious workers that always give incorrect labels, and avoid being detected as spammers (random or uniform) at the same time. It is also possible due to the workers who lack domain knowledge for labeling task.

(4) *Proper Worker*: after eliminating spammers, the remaining are proper workers. This type of workers have domain knowledge and provide high quality labels for items. They will follow the criteria of labeling task and take efforts to label the items. We define biased workers among the proper worker group. That means, a proper worker is either a biased worker or an un-biased worker.

Figure 3.5 gives examples of different types of spammers. Assume the label scale is from 1 to 6. The small circles represent the true labels, whereas the triangle signs in the figure show the labels given by the workers. X-axis in the figure represents different items, and y-axis is the label scale. Example of random spammer is given in figure 3.5(a), where worker randomly chooses labels from the scale for each item. Uniform worker shown in figure 3.5(b) presents label pattern as "$3, 4$". Whereas sloppy worker in figure 3.5(c) offers low quality labels, and no repeating label pattern is detected.

Both proper workers and sloppy workers are ethical workers, which are those who take time and efforts for the task and trying to give meaningful labels. Sloppy workers are possible to convert to random or uniform spammers. As an example, if an ethical worker from business school, who does not have programming knowledge, tries to grade students$'$ assignments from computer programming class, he might be a sloppy worker. However, after some time, he may be tired of the vain attempts, and he decides to give random grades. Now he is transformed into a random spammer.

Based on the classification of the graders, our goal is to detect and remove the random spammers, uniform spammers and sloppy workers, which will enable to improve the performance of final consensus results. After filtering the spammers, some new workers may be included for

replenishment.

3.3.2 Spam detection algorithm

In order to remove the different types of spammers from the workers group, spam detection algorithm proposed in [60] will be adjusted and adopted in our research.

(1) *Random Spammer Detection*: Random spammer is defined as the worker who has mean square error larger than a predefined threshold. Thus, for separating random spammers, the *RandomSep* function based on average squared error is used. By setting a threshold for *RandomSep* function for each worker, random spammers could be separated from other classes of workers.

$$RandomSep = \frac{\sum_{v \in V} \varepsilon_r^2}{|V|} \tag{3.4}$$

where ε_r is the error which represents the ordinal difference between a given label and the true label (or estimated true label). For example, assume workers are based on ordinal scale: $\{A, B, C, D, F\}$, which $A \to 5, B \to 4, C \to 3, D \to 2, F \to 1$. A worker provides C for an item, with the true label as A, thus $\varepsilon_r = -2$. $|V|$ is total number in the collection of all the labels offered by each worker.

Utilizing *RandomSep* to detect random spammers can be validated as following: according to Dawid and Skene [46], even though proper workers sometimes make errors, their error labels will be on the diagonal of true labels. For instance, there is a set of items, which with label scale as $\{A, B, C, D, F\}$, is graded through crowdsourcing, and true labels are already known. Comparison is then done between the observed labels and true labels. Assume there are totally 10 items with true labels as "A", and labeler i correctly answered 7 of them and mistakenly provides labels as "B" for the remaining three items. Thus i has the accuracy as 0.7, and error rate 0.3 for providing "B" to items with true label as "A". Similarly, for items with true grades as "B", the worker might have a 0.6 accuracy, and different error rates to give labels as "A" and "C" for the remaining items respectively. This is so called err on diagonal of true labels. Table 3.1 shows this proper worker example. The worker has high accuracy and usually provide wrong labels up or down one level of the truths even when they make errors. Different from proper workers, random spammers may give any labels regardless of the true grades such as F for an item whereas true label should be A.

TABLE 3.1

Example of proper worker with err on diagonal of true labels

Observed Labels		A	B	C	D	F
True Labels	A	**0.70**	0.30	0.00	0.00	0.00
	B	0.30	**0.60**	0.10	0.00	0.00
	C	0.00	0.00	**0.80**	0.20	0.00
	D	0.00	0.00	0.40	**0.50**	0.10
	F	0.00	0.00	0.00	0.30	**0.70**

(2) *Uniform Spammer Detection*: In order to detect uniform spammers, which present repeating labeling patterns, $UniformSep$ algorithm based on these repeating patterns is applied:

$$UniformSep = \sum_{s \in S} \frac{|s|^2 \cdot (f-1)^2 \cdot \varepsilon}{\theta} \tag{3.5}$$

where S is a collection of all possible n-tuples from available labels within a predefined range of lengths. For example, if the available labels set is: $\{1, 2, 3\}$, and predefined range of length is $\{2\}$. S then is $\{(1,2),(1,3),(2,3),(1,1),(2,2),(3,3),(2,1),(3,1),(3,2)\}$. $s \in S$ are matched within the ordered sequence of labels provided by each worker. $|s|$ is the length of s, and f is the frequency of s within the worker's labels. ε is the number of labels that are not same as the true labels within all matches of s.

Similar to random spammers, by setting up a threshold for $UniformSep$, uniform spammers can be classified. If a worker's $UniformSep$ is greater than the predefined threshold, then this worker is detected as uniform spammer. Otherwise, the worker does not belong to uniform spammer group. The $UniformSep$ used in this research has been tuned according to experimental heuristics, comparing to algorithm in [60]. Instead of using ε^2, ε is used here. By utilizing ε, we are able to set threshold as 1 for removing uniform spammers, whereas ε^2 requires changing the threshold value every time when running the simulated experiments. Since each worker is assumed to label in the range of 4 to 10 items, so the best lengths are $2, 3$, which means $LENGTHS = \{2, 3\}$. $\theta = 15|Reviews||LENGTHS|$. $|Reviews|$ is the number of items each worker reviewed.

(3) *Sloppy Worker*: : Sloppy workers are different from random and uniform spammers, they don't have fixed behavior patterns. After detecting and removing random and uniform spammers, it is much easier to separate sloppy workers from proper labelers. Since sloppy workers usually produce poor quality labels, it is reasonable to create a threshold for accuracy, which is the

percentage of correct answers given, for workers to filter them out. Labelers′ accuracy under the threshold are considered as sloppy workers. For instance, the threshold might be set as 0.6 to filter the sloppy workers. Labelers, with accuracy lower than 0.6, will be considered as sloppy workers and removed from the crowd.

Initially the approach proposed by Vuurens et al. [60] is designed for categorical (discrete) data type such as relevance judgement for IR-system. We have adjusted and tested with the continuous data, and proved it is applicable to numerical labeling scale tasks. Consider the example of grading on continuous scale from 0 to 100. To detect spammers by using the proposed algorithm, the labels $\in [0, 100]$ can be first converted to ordinal categorical scale, such as $\{A, B, C, D, F\}$. After removing the spammers (including sloppy workers) and adding new proper workers, original numerical labels could then be aggregated into consensus labels.

3.4 Reducing the number of workers

After removing the spammers, new workers are needed to be included for replenishment. In general, the number of new workers added has two different options:

(1) Same number of new workers be added as the removed workers. In this case, each item which need to be graded has a predefined number of workers (e.g. 5 labels each item).

(2) Define a threshold n for the number of workers required for each item, which is usually less than the initial number of workers per item. That means, every item should be labeled by at least n labelers. This approach gives the possibility of adding less workers after filtering spammers. Consider a labeling process that 10 workers are employed to evaluate each item initially. Let the item with least number of workers as i after removing spammers, and the amount of worker for i is 6. Threshold n was set as 4. In this case, no new labeler is required to be included. Comparing to the first option, in which at least 4 new workers have to be added, setting threshold approach need less new labelers.

In crowdsourcing systems, balance should be maintained between the accuracy of results and the cost for labeling, which means get as accurate answers as possible within the budget for labeling. The requirement for the accuracy of results may vary with situations:

(a) The budget is fixed and limited, and get labels with as less cost as possible. The accuracy of the labels should also be as accurate as possible, but a certain range of errors are allowed. As

Algorithm 3.1 Reducing the number of workers by setting threshold for labels for each item

Input: A set of items $S = \{s_1, s_2, , s_n\}$, initial graders set U, and initial number of labels for each item as m, the set R which consists of the remaining number r_i of labels for each item s_i, and the threshold t for the number of labels required for each item after removing spammers.

Output: The number of new proper workers x.

```
{Initialization}: set x = 0, and G' = φ
for each s_i ∈ S do
    Set g_i = t − r_i
    if g_i > 0 then
        G' = G' ∪ {g_i}
    end if
end for
while G' ≠ φ do
    Let increment = max(g'_i), g'_i ∈ G'
    x = x + increment
    update G':
        Random selection one g'_i == increment, let G' = G' − {g'_i}
        Random Select (m×|S|)/|U| items from G', denoted as G'_j
        for each selected g'_j ∈ G'_j do
            Let g'_j = g'_j − 1
            if g'_j = 0 then
                G' = G' − {g'_j}
            end if
        end for
    end update
end while
```

an example, grading students' homework in computer programming class.

(b) Get as close as possible to true labels regardless of the cost. In this case, errors are intolerable which means a small error could cause the disaster such as medical disease diagnose process.

Most crowdsourcing cases fall into first category since reducing the cost for human intelligence tasks is the most outstanding feature of crowdsourcing. Thus it is necessary to analyze the possibility of adding fewer new labelers to reduce the cost. Defining the threshold for the number of labels for an item gives the chance to reduce the number of workers. The formalization of the worker reduction problem is as follows:

Assume the initial amount of workers who evaluated each item is m, S is the set of n items ($S = \{s_1, s_2, ..., s_n\}$), and U is initial workers set. $|S|$ is the number of total items. The threshold defined for the number of labels of every item is t. Thus each worker is able to evaluate $(m \times |S|)/|U|$ items. After removing spammers, the remaining number of labels for item s_i is $r_i \in R$ ($R = \{r_1, r_2, ..., r_n\}$).

$$g_i = \begin{cases} 0, \quad if \quad (t - r_i) \leq 0; \\ t - r_i, \quad otherwise \end{cases} \qquad i = 1, 2, ..., n \tag{3.6}$$

where g_i is the number of required labels from new workers. Assume G' is the set consisting of $g_i \neq 0$, $i = 1, 2, ..., n$. The number of new workers x need to be added is calculated in the following process:

(1) Initialize $x = 0$;

(2) If $G' \neq \phi$, exit the process. Otherwise, make $increment = max(g'_i)$, $g'_i \in G'$ and $x = x + increment$;

(3) Update set G': a) Remove g'_i which equals to $increment$ from G'. If more than one g'_i is same as $increment$, random select one and remove it; b) Randomly select $(m \times |S|)/|U|$ items from G', update $g'_i = g'_i - 1$ for all selected $g'_i \in G'$. If updated $g'_i = 0$, remove this item from G'. Repeat this step for increment times;

(4) Repeat step 2 and 3 until $G' = \phi$ to exit the process. Finally, we could get the number of new workers get included equals to x.

Algorithm 3.1 shows this process to optimize the number of workers.

3.5 Integrated approach for aggregating crowdsourced labels

The proposed Spam Removing and De-biasing Framework (SRDF) integrates the bias detection and spam filtering algorithms discussed in Section 3.2 and Section 3.3 in the process of aggregating the answers collected from repeated labeling. The purpose is to achieve consensus results with high quality. The aggregating approach used in this chapter can be grouped into two categories:

(1) Vancouver algorithm: It is an extension of the iterative learning algorithm developed in [38], and more discussions about it are given in [7]. To the purpose of easier referring, we use the name proposed in [7] which is "Vancouver" for this approach. The Vancouver algorithm leans the variance of worker provided labels, by comparing the observed labels to the estimated true labels. The inversion of the variance is then utilized to weight the labels provided by each worker in order to obtain the consensus results.

(2) Baseline algorithms including a) average, and b) median approach. Baseline algorithm is a simple, yet reasonable method that is applied to establish minimum expected performance on a data set [116]. The baseline algorithms fail to take into account workers' reliability while aggregating the observed labels. They assume that each worker has the same trustworthiness.

We present the brief introduction of these two types of aggregating algorithms in Section 3.5.1. Section 3.5.2 then gives details about the integrated SRDF approach to obtain the desired consensus labels for items.

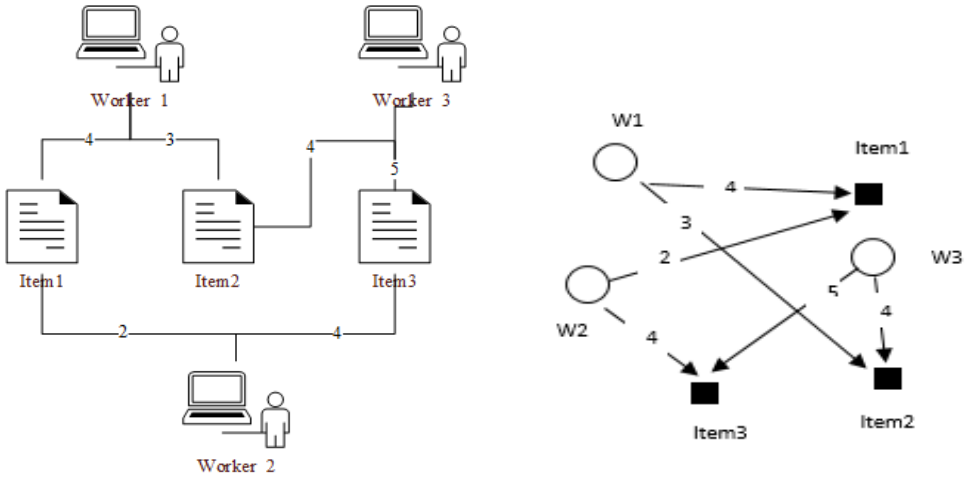
3.5.1 The aggregating algorithms

3.5.1.1 Basics of Vancouver algorithm

Vancouver algorithm measures worker's labeling accuracy, and takes into account the weight of each worker while aggregating collected labels, we are extending this approach for our framework. The weight of a worker reflects the reliability of him, and higher weight denotes that a worker's label has the larger possibility to be chosen as estimated truth. In Vancouver algorithm, each worker's labeling accuracy is measured by comparing the labels assigned by him, with the labels given by other workers to the same submission [7]. It gives more weight to the input of workers with higher measured accuracy. In particular, the Vancouver algorithm suggests that worker's accuracy can be measured through his variance, which is calculated by comparing the observed labels, which is offered by workers, to the consensus labels. For example, if the calculated consensus labels for items s_1, s_2, s_3 are 75, 80, 90, on the scale from 1 to 100, whereas the labels provided by worker i for these items are: 80, 90, 85. Then the variance for this worker is: $[(80-75)^2 + (90-80)^2 + (85-90)^2]/3 = 50$. In aggregating process, the labels provided by worker i could be weighed in proportion to $1/v_i$, where v_i is worker i's variance.

The algorithm proceeds in iterative fashion, using consensus labels to estimate the labeling variance of each worker, and using the information on each worker variance to compute more precise consensus results. The approach proposed that the review relations between workers and items could be seen as a graph. The nodes are the workers and the items from the labeling task. Figure 3.6(a) shows a labeling network represents the review relation between the workers and items. Each worker is responsible for multiple items, and every item is also labeled by multiple workers. For example, Worker1 evaluated two items: Item1 and Item2. Whereas Item3 was labeled by Worker2 and Worker3. Figure 3.6(b) transforms the network into a graph presentation, with circle nodes represent workers and square nodes are items.

The algorithm may be formalized as: We denote by U the set of workers, and by S the set of items to be labeled. We let $G = (T, E)$ be the graph encoding the review relation, where $T = S \cup U$ and $S \cap U = \phi$, and where $(i, j) \in E$ iff j reviewed i; for $(i, j) \in E$, we let g_{ij} be the labels assigned by j to i. We denote by ∂t the 1-neighborhood of a node $t \in T$: a) If t is worker



(a) Labeling network: numbers on connections are the labels given by workers

(b) Review relation graph: O represents workers (W: worker), ■ are items, numbers on arrows are labels provided by workers

Figure 3.6: Graph representation of the review relation between workers and items

node, items evaluated by t are ∂t, and b) If t is item node, workers who graded t are ∂t.

The algorithm proceeds by updating estimates of:

(1) variance v_j of worker $j \in U$,

(2) c_i of the consensus label of item $i \in S$, and

(3) v_i of the variance of item i with which c_i is known.

To produce these estimates (v_j, c_i, v_i), the algorithm relies on messages $M = \{m_1, m_2, ...\}$, with each message $m = (l, x, v)$ consisting of a source $l \in S \cup U$, a value x, and a variance v. We denote by M_i, M_j the lists of messages associated with item $i \in T$ or worker $j \in U$. Given a set M of messages, we indicate by:

$$E(M) = \frac{\sum_{(l,x,v)\in M} {}^{x}/_{v}}{\sum_{(l,x,v)\in M} {}^{1}/_{v}} \tag{3.7}$$

$$var(M) = \left(\sum_{(l,x,v)\in M} \frac{1}{v} \right)^{-1} \tag{3.8}$$

The best estimator $E(M)$ we can obtain from M, and its variance $var(M)$.

Take Figure 3.6(b) as an example to illustrate the algorithm. G is the graph showed in the figure. The set of workers $U = \{W_1, W_2, W_3\}$, and item set $S = \{Item_1, Item_2, Item_3\}$. In order to estimate variance for each worker, consensus label and variance for each item, a list of messages are created. The messages can be divided into two categories:

(1) Messages for workers: each worker is associated with a set of messages, with one message for each item he graded. For instance, Worker1 should have a set of messages M_1 which consist of two separate messages. Each message m corresponds to the item evaluated by W_1: $m = (i, \hat{g}_i, v_i)$, where i represents submission i which is graded by W_1, $\hat{g}_i$ is the consensus label calculated for item i, and v_i is the estimated variance of item i. Thus $M_1 = \{(1, \hat{g}_1, v_1), (2, \hat{g}_2, v_2)\}$.

(2) Messages for items: each item is associated with a set of messages, with one message for every worker who evaluated it. For example, Item1 would have a set of Messages M_1' which contains two different messages. Every message m' correlates to the worker who labeled Item1: $m' = (j, g_{1j}, v_j)$, where j is the worker who evaluated Item1, g_{1j} is the label given by worker j to Item1, and v_j is the variance of worker j. Thus, $M_1' = \{(1, 4, v_1), (2, 2, v_2)\}$.

The details of the process are shown in Algorithm 3.2 [7]. The input is the graph which represents the review relations between workers and items, and the labels given by the workers for each item (observed labels). The output of the algorithm is an estimation of the true label (also called consensus labels) for every item. K is the number of iterations of the process, which should be large enough until the worker's variance results converge. Lines 2-4 initialize the messages associated with items using the label assigned by the workers, and a constant variance as initial value. All the workers$'$ variance in item message lists are initialized with 1. Lines 7-9 initialize the messages correlated to workers as ϕ. Lines 10-14 propagate the best estimate available on item labels and variances, from item to labelers who graded them. It updates the messages list for every worker. Similarly, lines 16-23 propagate the label for the item from labeler, and the worker's variance, from workers to items reviewed by them. These steps are used to re-estimate the variables in messages list for each item. The final aggregation in lines 26-28 aggregates the information from workers, including labels and variance, into the estimation of labels (consensus labels) for items.

As an example, the process to calculate the consensus labels for items in Figure 3.6(b) is as follows:

(1) Initialize the messages list M_i associated with every item i: $M_1 = \{(1, 4, 1), (2, 2, 1)\}, M_2 = \{(1, 3, 1), (3, 4, 1)\}, M_3 = \{(2, 4, 1), (3, 5, 1)\}$.

(2) Update messages list M_j associated with every worker with information in M_i: Worker1 evaluated Item1 and Item2, thus his messages list should have two m's. M_1 for Worker1 equals to $\left\{\left(1, 4 \times \frac{1}{1+1} + 2 \times \frac{1}{1+1}, \frac{(4-3)^2+(2-3)^2}{2}\right), \left(2, 3 \times \frac{1}{1+1} + 4 \times \frac{1}{1+1}, \frac{(3-3.5)^2+(4-3.5)^2}{2}\right)\right\} = \{(1, 3, 1), (2, 3.5, 0.25)\}$. Similarly, M_2 and M_3 could be calculated for Worker2 and Worker3.

Algorithm 3.2 The basic Vancouver algorithm [7]

Input: A review graph $G = ((S \cup U), E)$ such that $|\partial t| > 1$ for all $t \in S \cup U$, along with $\{g_{ij}\}_{(i,j)\in E}$, and number of iterations $K > 0$.
Output: Estimates $\hat{q}_i$ for $i \in S$

{Initialization}
for all $i \in S$ **do**
    $M_i := \{(j, g_{ij}, 1) | (i, j) \in E\}$
end for
for $iterationk = 1, 2, ..., K$ **do**
    {Propagation from items}
    for all $j \in U$ **do**
        $M_j := \phi$
    end for
    for all $i \in S$ **do**
        for all $j \in \partial i$ **do**
            Let $M_{-j} = \{(j' \in M_i | j' \neq j)\}$ in $M_j := M_j \cup ((i, E(M_{-j}), var(M_{-j}))$
        end for
    end for
    {Propagation from workers}
    for all $i \in S$ **do**
        $M_i := \phi$
    end for
    for all $j \in U$ **do**
        for all $i \in \partial j$ **do**
            Let $M_{-i} = \{(i', (x - g_{i'j})^2, v) | (i', x, v) \in M_j\}$ in $M_i := M_i \cup (j, g_{ij}, E(M_{-j}))$
        end for
    end for
end for
{Final Aggregation}
for all $i \in S$ **do**
    $\hat{q}_i := E(M_i)$
end for

(3) Update messages list M_i associated with each item with calculated M_j in step 2: For Item1, $M_1 = \left\{\left(1, 4, \frac{(4-3)^2+(3-3.5)^2}{2}\right), \left(2, 2, \frac{(2-3)^2+(4-4.5)^2}{2}\right)\right\} = \{(1, 4, 0.625), (2, 2, 0.625)\}$. M_2 and M_3 can also be calculated for Item2 and Item3 in same way.

(4) Repeat step 2 and 3 K times so that the results for v_i and v_j converge (no longer change).

(5) Finally, estimated true labels are able to be obtained for item i through calculating $E(M_i)$ (equation 3.7).

3.5.1.2 Baseline aggregating algorithms

Baselines models are provided to compare with proposed methodology. There are two prevalent baseline aggregating algorithms used in this research: average algorithm (AVG), and median algorithm (MED). AVG is a very basic algorithm while aggregating the observed labels into consensus labels. AVG algorithm is a non-iterative way to estimate final labels, and it does not take into account the workers$'$ reliability or expertise. It just simply averages the labels each item received to

get the consensus labels. As an example, if the item i received labels $\{80, 75, 90, 88\}$ from different workers through crowdsourcing systems, then the consensus label obtained for the item through AVG is: $\frac{(80+75+90+88)}{4} = 83.25$.

Median algorithm take the median value among the observed labels for one item as consensus result. Assume item i obtained labels from crowd as $\{90, 85, 60, 82, 90\}$, then the consensus label is 85. If the observed labels are $\{90, 85, 60, 82\}$, the estimated label for the item is calculated as $\frac{(85+82)}{2} = 83.5$.

The reason to use both AVG and median baseline approaches is to handle different distributions of labels obtained from crowd workers: AVG is more applicable when observed labels have normal number distributions, which has a low amount of outliers. Whereas median algorithm is more suitable for skewed observed label distributions.

3.5.2 Spam removing and de-biasing framework (SRDF)

In order to get consensus labels as close as possible to labels given by experts, this research develops a crowdsourcing framework SRDF (Spam Removing and De-biasing Framework), which combines the spam detection and removal algorithm with the proposed bias correction approach. Compared to the existing work, which fails to distinguish the correctable errors, which are made by biased workers, from the true errors, which are shown within the labels from spammers, the proposed SRDF recognize spam and biased workers separately. Additionally, we propose to reduce the number of workers by setting a threshold for the minimum amount of labels required for each item. The SRDF framework is shown in Figure 3.7, and the details of the approach is described as follows:

(1) *Items distributed to workers for labeling*: Randomly select workers for the labeling task. Each item is required to be reviewed by a specified number of workers (e.g. 6 workers per item).

(2) *Apply spam detection algorithm*: Spam detection algorithm is used to analyze the observed labels collected from step 1 to detect random and uniform spammers. The workers recognized as spammers are then rejected and their labels are removed from the observed label set. After filtering these two types of spammers, and before removing sloppy workers, de-biasing algorithm is utilized to correct the biased labels. It is designed to prevent mistakenly removing useful biased workers as sloppy workers. Consider the following case: Assume worker w provides grades $\{A, B, B, A, B\}$ for a subset of items, whereas the true labels should be $\{B, C, C, B, C\}$.

Without de-biasing process, the workers will be removed as sloppy worker since his accuracy is 0. However, by correcting the given labels one level up, we could obtain high quality results from worker w. Finally, sloppy workers are removed according to their accuracy.

(3) *Add new workers*: In order to replenish the rejected workers, some new reviewers need to be incorporated into the worker group. New reviewers are added according to different settings of required number of workers. If the new workers are also spammers recognized by the spam detection algorithm, keep on adding until enough proper workers.

(4) *De-bias workers' labels*: Bias detection approach is utilized to recognize bias patterns for the workers. Bias is then reduced from the labels by subtracting the median $diff(t_i, o_i)$ from the observed labels for the biased workers.

(5) *Get consensus labels*: Finally, AVG and Vancouver algorithm are applied to get the consensus labels from aggregating the de-biased results obtained from step 4.

3.6 Experimental analysis for SRDF methodology

This section presents experimental analysis for the proposed SRDF approach on the simulated data sets. We show the influence of spam worker filtering algorithm and bias detection method on the consensus results separately, and then the integrated approach SRDF which combines these two approaches is tested and presented on the synthetic dataset. Standard deviation, correlation coefficient and root mean square error are used as the metrics for evaluating the experimental results. Section 3.6.1 presents the datasets and evaluation metrics for conducting the experiments. All the results performed on the dataset and their analysis are presented in Section 3.6.2.

3.6.1 Experimental setup

3.6.1.1 Evaluation metrics

There are many different measures, which are used for the labeling problem with numerical scales, such as precision, recall, and Pearsons correlation, etc. [5, 7, 60] In our research, standard deviation (σ), correlation coefficient (ρ) and root mean square error ($RMSE$) [7, 117] are used to measure the performance of the proposed frameworks. These measurements are able to evaluate how close the estimated labels compare to the true labels on the continuous scale, and they are

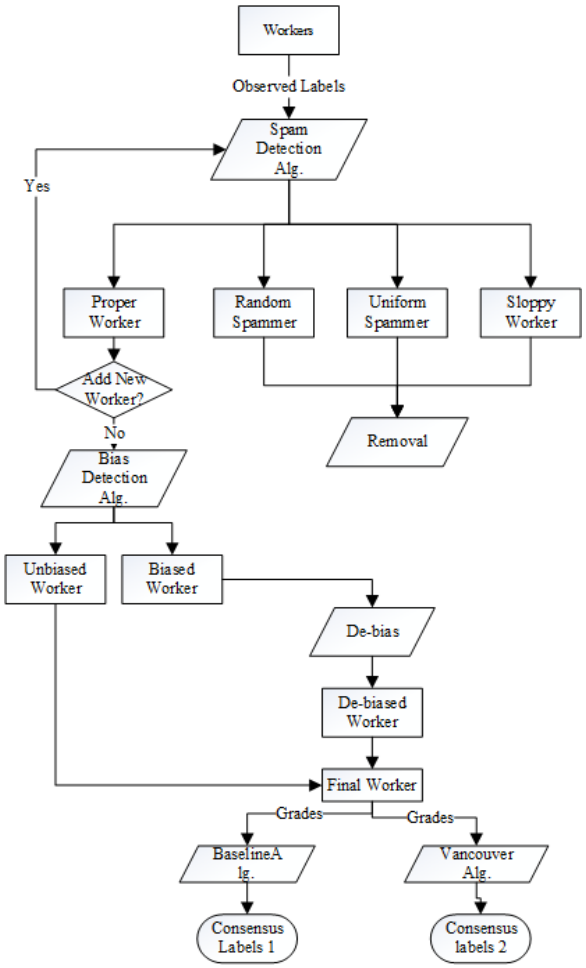


Figure 3.7: Spam removing & de-biasing framework (SRDF)

commonly used in tasks such as grading students′ submissions [5, 7]. They can be calculated as follows:

$$\sigma = \sqrt{E[(\hat{Y} - \mu)^2]} \tag{3.9}$$

$$\mu = E(\hat{Y}) \tag{3.10}$$

$$\rho_{Y,\hat{Y}} = \frac{COV(Y, \hat{Y})}{\rho_Y \rho_{\hat{Y}}} \tag{3.11}$$

$$COV(Y, \hat{Y}) = E((Y - \mu_Y)(\hat{Y} - \mu_{\hat{Y}})) \tag{3.12}$$

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n} (\hat{y}_t - y)^2}{n}} \tag{3.13}$$

where $E(\hat{Y})$ is the expected value of $\hat{Y}$, and $\hat{Y}$ is consensus labels (estimated labels). $\rho_{Y,\hat{Y}}$ is the correlation coefficient of consensus labels $\hat{Y}$ with true labels Y. $COV(Y, \hat{Y})$ is the covariance of true

values and consensus labels. In 3.13, $\hat{y}_t$ is the consensus label for item t, and y is the true value for t.

3.6.1.2 Description of datasets

To test the proposed methodology, simulations about the labeling process were used to create the simulated data set. The way of conducting the simulations are derived from the patterns and discoveries of the work in [46] and [10], which are both based on real world applications and data. There are totally 50 workers and 50 items generated for the experiments. The simulation process is as follows:

(1) Generate a set of items: The item set which needs to be labeled with true quality based on discrete label scale: $\{A, B, C, D, F\}$, which is same as $\{5, 4, 3, 2, 1\}$. The categorical true quality c_i of item i we assumed was normal distributed with standard deviation of 1.0. Randomly generate a number r from the dustribution, if $r \in [\mu - 2\sigma, \mu + 2\sigma]$, we assume c_i uniformly distributed on $\{B, C, D\}$. If $r < \mu - 2\sigma$, then $c_i = F$, otherwise $c_i = A$.

After getting categorical true label, the continuous true label qi which is on $[50, 100]$ scale needs to be created for each item corresponding to c_i. The relationship between c_i and q_i is:

$$q_i \in \begin{cases} [90, 100], & \text{if} \quad c_i = A \\ [80, 90), & \text{if} \quad c_i = B \\ [70, 80), & \text{if} \quad c_i = C \\ [60, 70), & \text{if} \quad c_i = D \\ [50, 60), & \text{if} \quad c_i = F \end{cases} \tag{3.14}$$

$$q_i = 55 + (c_i - 1) \cdot 10 + \Delta_i \tag{3.15}$$

Equation 3.15 shows the way to get q_i for each item, in which $c_i \in \{5, 4, 3, 2, 1\}$ while calculating, and Δ_i is normal distributed with $\mu = 0.0$, and within the range of $[-5, 5]$. There are totally 50 submissions created.

(2) Generate a set of workers: Assuming the initial fractions of random spammers, uniform spammers and sloppy workers are all equals to 0.1. Among proper workers, the ratio of biased workers is 0.2. Proper workers′ accuracy *acc* should be between 0.5 and 0.9, and sloppy worker's

$acc \in [0.2, 0.4]$. Both of them are assumed to be normal distributed. We consider 50 workers for the simulation, with each worker labeling a required number of items.

While labeling, both categorical and continuous label should be provided for each item. Suppose d_{ij} and g_{ij} is the categorical and continuous label for item i from worker j, respectively. For random spammer, d_{ij} is randomly selected from $\{A, B, C, D, F\}$. Uniform spammer is set to have a fixed grading pattern, which means repeatedly gives same labels. To simulate sloppy worker, we randomly give label d_{ij} one level up or down of c_i. For example, if c_i is B, then randomly select from $\{A, C\}$ as d_{ij}. If the biased worker err while grading, d_{ij} has to be:

$$d_{ij} = \begin{cases} c_i - 1, & \text{if} \quad j \textit{ has negative bias pattern} \\ c_i + 1, & \text{if} \quad j \textit{ has positive bias pattern} \end{cases} \tag{3.16}$$

The continuous label g_{ij} is calculated based on d_{ij}:

$$g_{ij} = 55 + (d_{ij} - 1) \cdot 10 + \Delta_{ij} \tag{3.17}$$

where Δ_{ij} has normal distribution with mean 0 and standard deviation 1, and it should be in the range $[-5, 5]$. However, if the biased worker's d_{ij} is same as c_i, it is necessary to make sure the generated g_{ij} greater than q_i if worker j has positive bias pattern, or g_{ij} less than $q_i$if negative.

3.6.2 Experimental results and analysis

The simulation process has an iteration of 100 runs to get more general and convincing results on all the three evaluation metrics. We assume initially each submission is reviewed by 6 graders. Section 3.6.2.1 presents the experiments which explore the influences of bias detection algorithm with different number of workers for each item. Vancouver aggregating approach is utilized to obtain the consensus results. In other words, we detect the biased workers and correct their labels using the proposed bias detection algorithm. The corrected labels along with other remaining observed labels are aggregated to obtain consensus labels by using Vancouver algorithm. This framework is denoted as MVD (Methodology with Vancouver & De-biasing). The comparisons of the results by applying MVD and other baseline approaches are also given in this section. Section 3.6.2.2 presents the influences of different types of spam workers on the quality of obtained consensus labels. The experimental results for SRDF are presented in Section 3.6.2.3.

TABLE 3.2

Results of metrics (ρ, σ, and $RMSE$) for different number of workers with MVD framework: N_w represents the number of workers, k is the shape factor of Gamma distribution ($k = 2, 3$).

MVD	k=2			k=3		
	ρ	σ	$RMSE$	ρ	σ	$RMSE$
$N_w = 4$	0.94	0.29	0.31	0.80	0.74	0.77
$N_w = 5$	0.96	0.21	0.22	0.86	0.57	0.62
$N_w = 6$	0.98	0.17	0.18	0.92	0.42	0.52
$N_w = 7$	0.99	0.12	0.13	0.95	0.31	0.47
$N_w = 8$	0.99	0.10	0.11	0.96	0.27	0.41

3.6.2.1 Experimental results for MVD and baseline models

Four different models for estimating true labels are analyzed here, and they are denoted as: AVG, MED, BASIC, and MVD.

(1) AVG model represents the average model. It uses the AVG aggregating algorithm to estimate the consensus labels.

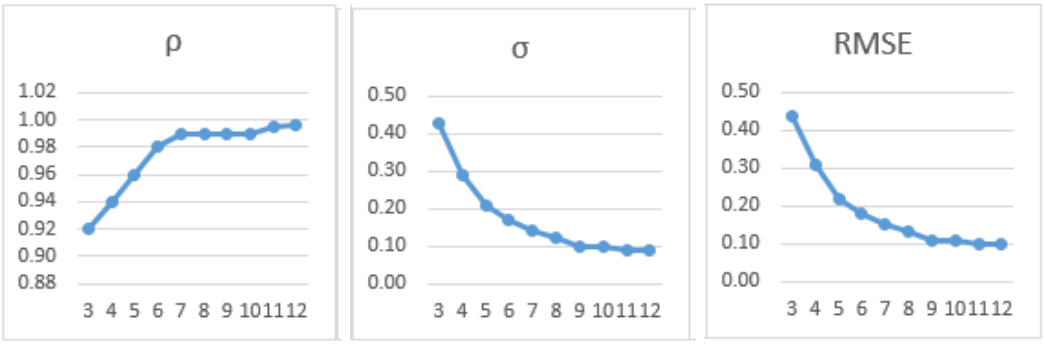
(2) MED model represents median model. Median algorithm is utilized to aggregate the observed labels into consensus labels in this model.

(3) BASIC model utilizes Vancouver algorithm as aggregating algorithm, however, without applying de-biasing process. BASIC is essentially the Vancouver approach used in [7].

(4) MVD model stands for method with bias detection and correction, and Vancouver as aggregating algorithm.

(I) Experimental results with different number of workers for each item on MVD

When crowdsourcing the labeling tasks, each item is required to be evaluated by multiple workers. The common number of workers employed per item in much research [5, 7, 60, 81] is around 5 or 6. In order to find out the most appropriate number of workers in our MVD approach, we conduct experiments with different number of labelers. The synthetic data set is simulated for 100 runs. Table 3.2 gives the results for three evaluation metrics ρ, σ, and $RMSE$ by applying MVD framework. Five different settings for the number of workers are tested: 4, 5, 6, 8, and 10. Three metrics (ρ, σ, and $RMSE$) are reported as the average over the 100 runs. To give better illustration about the influence of workers$'$ number per item to final results, we apply the MVD approach with more different cases. The results are shown in Figure 3.8. The worker size varies from 3 to 12 for each



(a) Correlation coefficient (b) Standard deviation (c) Root mean square error

Figure 3.8: Results for metrics with different number of workers: x-axis is number of workers/item (N_w), y-axis represents the value of calculated corresponding metric.

item. It is observed that before the point with $N_w = 6$, the performance of all the three measures changes greatly. After this point, the values of the metrics varies slightly, and almost remains stable with $N_w \geq 7$. While having 11 to 12 worker review each item, ρ approaches to 1 (0.996). The ranges in which the evaluation metrics vary are $[0.92, 0.996]$, $[0.09, 0.43]$, and $[0.10, 0.44]$ for ρ, σ and $RMSE$ respectively.

Best results could be obtained by setting the number of workers as 10 in Table 3.2, which is the maximum N_w among all the settings. However, more workers result in more cost for employing them. In order to compare our methodology with basic Vancouver algorithm, we require each item be reviewed by 6 workers as Alfaro and Shavlovsky proposed in [7]. In addition, $N_w = 6$ is able to yield acceptable accuracy as shown in Table 3.2 and Figure 3.8.

(II) Experimental results for MVD & baseline models

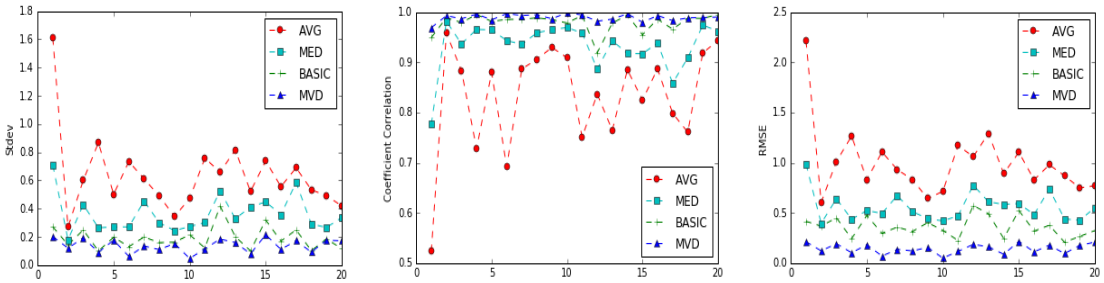
In this section, AVG model and MED model are worked as baseline models, which are simple but reasonable models to establish minimum expected performance on data sets [116]. The synthetic data is simulated for 100 runs, and the evaluation metrics are then reported as the average over these runs as before.

Figure 3.9 shows the evaluation metrics calculated for the first 20 simulation runs, with shape factor k=2 of the Gamma distribution, for different models. Figure 3.9(a) presents the results of σ, ρ, and $RMSE$ for AVG, MED, BASIC and MVD models. The line with small "o" is the outcome for AVG model, and results for MED model are shown on square marked line. The green line with "+" sign and the blue line with triangle marker represent the results for BASIC and MVD model separately. To show the difference between BASIC and MVD model, we give more detailed

TABLE 3.3

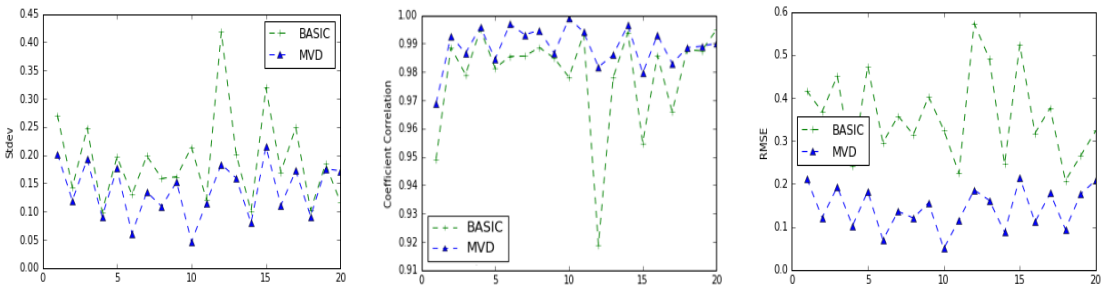
Experimental results of ρ, σ, and $RMSE$ for different models for 100 runs (N_w: number of workers per item)

$N_w = 6$	k=2			k=3		
	ρ	σ	$RMSE$	ρ	σ	$RMSE$
AVG	0.812	0.690	0.697	0.619	1.255	1.269
MED	0.890	0.496	0.522	0.779	0.819	1.106
BASIC	0.977	0.201	0.207	0.913	0.425	0.431
MVD	**0.982**	**0.173**	**0.179**	**0.920**	**0.413**	**0.410**



(I) Standard Deviation (II) Correlation Coefficient (III) RMSE

(a) Evaluation metrics results on synthetic data for four models for 20 runs



(I) Standard Deviation (II) Correlation Coefficient (III) RMSE

(b) Evaluation metrics results on synthetic data for BASIC & MVD for 20 runs

Figure 3.9: Standard deviation, Correlation coefficient and RMSE for different models on synthetic data for 20 runs (k=2)

figures in Figure 3.9(b).

From Figure 3.9(a), we could see that AVG model gave the worst results for all the three evaluation metrics, whereas MVD has the best results among the four different models. The ordered sequence of the accuracy for the models based on the metrics are: $MVD > BASIC > MED > AVG$. To further investigate the improvement of MVD model compared to other models, we run 100 simulations on all the four models. The averaged results are shown in Table 3.3.

The experimental results presented in Table 3.3 are consistent with Figure 3.9, in which MVD model are the most outstanding among all the models. All the three metrics we selected are improved

TABLE 3.4

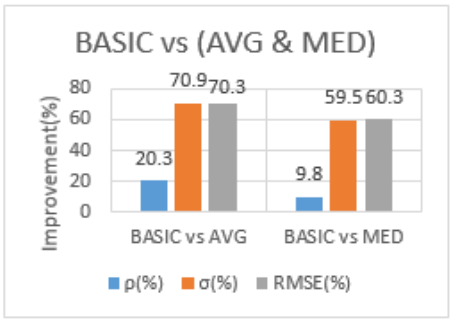
Results for BASIC and MVD model with increased biased workers ($k = 2$, and N_w is the number of workers per item)

$N_w = 6$		Num. of biased workers			
		9	12	24	28
ρ	BASIC	0.984	0.977	0.968	0.971
	MV	0.988	0.982	0.99	0.995
	Impr(%)	0.40%	0.30%	2.27%	2.47%
σ	BASIC	0.177	0.225	0.223	0.246
	MVD	0.149	0.193	0.137	0.0997
	Impr(%)	15.82%	14.22%	38.57%	59.47%
$RMSE$	BASIC	0.179	0.225	0.438	0.451
	MVD	0.153	0.197	0.154	0.1
	Impr(%)	14.53%	12.44%	64.84%	77.83%

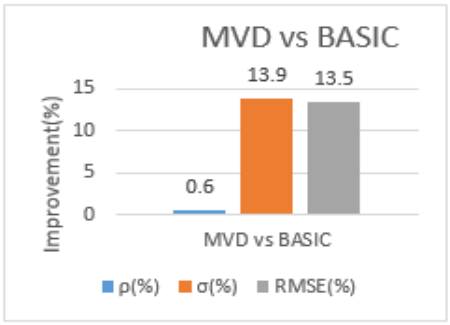
for the consensus results after applying BASIC model compared to baseline models – AVG and MED model. After de-biasing, which means applying MVD, there is further improvement compared to BASIC approach. As an example, if standards deviation has 71% improvement comparing BASIC with AVG, and 14% improvement comparing MVD with BASIC, then approximate 75% improvement is given comparing MVD with AVG model. The correlation coefficient, standard deviation, and RMSE can be up to 0.982, 0.173 and 0.179 by applying MVD method.

Figure 3.10 presents the percentages of improvement for the metrics by comparing the results of the four models, which is calculated from Table 3.3. Figure 3.10(a)(I) and Figure 3.10(b)(a) show the enhancement obtained for the evaluation metrics from comparisons of BASIC with AVG model, and BASIC with MED model, with shape factor equals to 2 and 3. In our experiment, we are able to improve ρ up to around 20%, reduce the σ by 70%, and lower the $RMSE$ by 69% by utilizing BASIC approach, in contrast to baseline models. Although there is only small improvement (0.6%) while comparing MVD with BASIC method, we could reduce error on $RMSE$ by 15%, and σ by around 16%.

By taking into account workers′ bias pattern, we could obtain significant gains in accuracy. To prove the importance of de-biasing the workers, we increase the number of biased workers in our data set. Table 3.4 shows the results of three selected metrics results and percentage of improvement comparing MVD with BASIC approach. As results presented, the more biased workers, the better improvement we could get by using the MVD model. For example, by having more biased workers, we are able to reduce the RMSE from 15% to 78%. The reason is that more workers with a bias pattern means less of them make random error, and as a result, more accurate labels could be obtained after de-biasing the workers.

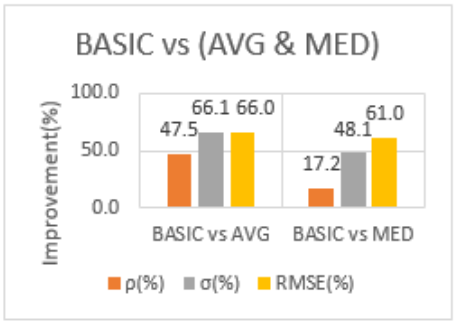


(I) Improvement of Results Comparing (BASIC vs AVG) & (BASIC vs MED)

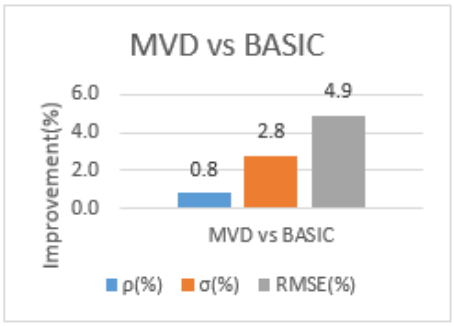


(II) Improvement of Results Comparing MVD vs BASIC

(a) Percentage of improvement by comparing results of different models with $k = 2$



(I) Improvement of Results Comparing (BASIC vs AVG) & (BASIC vs MED)



(II) Improvement of Results Comparing MVD vs BASIC

(b) Percentage of improvement by comparing results of different models with $k = 3$

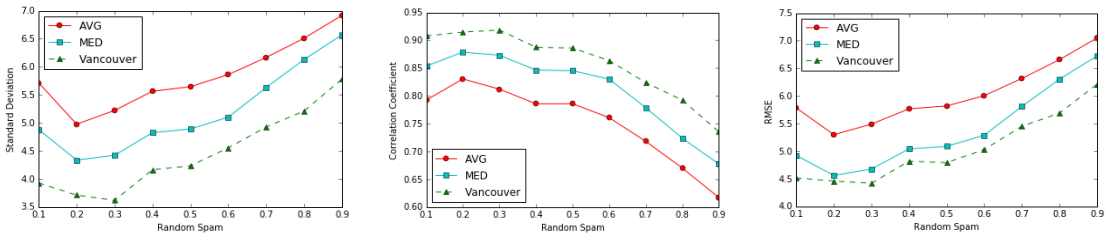
Figure 3.10: Improvement (%) of results comparing different models

3.6.2.2 Experimental results for SRDF methodology

In Section 3.6.2.1, the Vancouver algorithm integrated with bias detection approach, which is MVD, has been analyzed. Here, the experimental results is presented, for combining both spam removing and bias detection approaches into Vancouver algorithm, which is proposed SRDF methodology. The simulation process has an iteration of 100 runs to get more general and convincing results on all the three evaluation metrics. We assume initially each item was reviewed by 6 workers.

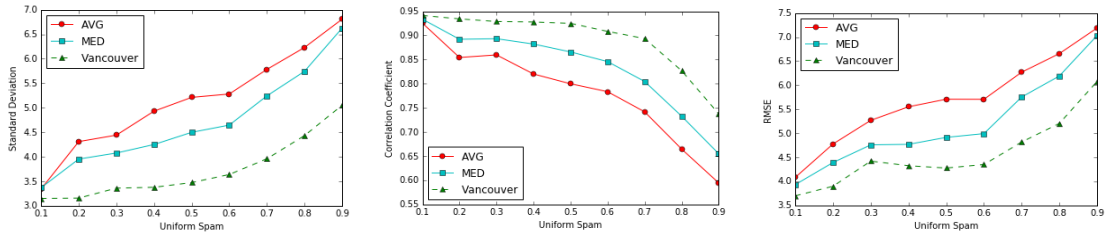
(I) Influences of different types of spammers

To check the impact of different types of spammers to the final consensus labels, we change the proportion of the spammer groups. For inspecting random spammers, we set the ratios of uniform spammer and sloppy worker both to 0. Increasing the proportion of random spammers from 0.1 to 0.9 with 10% increasing rate. In order to check uniform spammers and sloppy workers, similar processes have been conducted. The results are presented in Figure 3.11.



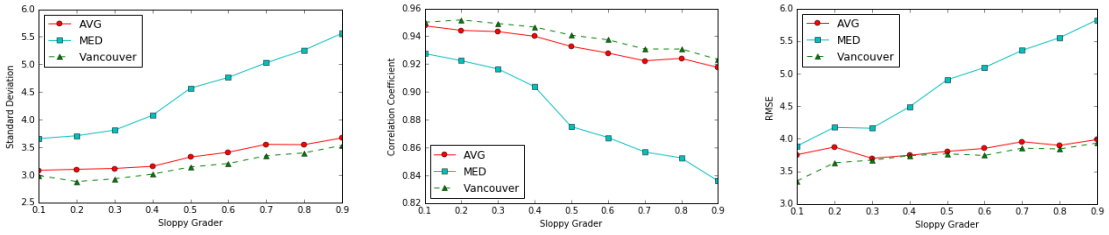
(I) Standard Deviation (II) Correlation Coefficient (III) RMSE

(a) Impact of random spammers on consensus labels



(I) Standard Deviation (II) Correlation Coefficient (III) RMSE

(b) Impact of uniform spammers on consensus labels



(I) Standard Deviation (II) Correlation Coefficient (III) RMSE

(c) Impact of sloppy workers on consensus labels

Figure 3.11: Improvement (%) of results comparing different models

Figure 3.11(a) represents the calculated metrics while increasing the percentage of random spammers. Two solid lines one with small circles and the other with squares show σ, ρ, and $RMSE$ results for AVG and MED model respectively. The dotted line with triangles represents Vancouver approach. The performance of all these three models decreases with the increasing population of random spammers. It could be seen that the model with Vancouver algorithm gives most accurate results compared to the baseline models, due to the fact that it takes into account the weight of each worker. Higher value on correlation coefficient means better accuracy, which is different from standard deviation and $RMSE$. Thus ρ calculated on Vancouver approach is larger than AVG and MED. The impact of uniform spammers and sloppy workers are quite similar to random spammers: the more spammers included, the worse performance of the models.

(II) Analysis of changing the order of spam filtering and de-biasing process

TABLE 3.5

Results for different orders of spam filtering and de-biasing components

Order of Two Components	σ	ρ	$RMSE$	# Proper Workers Added
Spam Filtering + De-biasing	2.06	0.97	2.21	21
De-biasing + Spam Filtering	3.35	0.94	3.48	13
Optimal Order	**2.01**	**0.97**	**2.15**	**15**

To test the effect of changing the order of utilizing spam filtering and de-biasing algorithm, simulations were run for the following proportion settings: Random spammers = 0.1, Uniform spammers = 0.1, Sloppy worker = 0.1, and Biased worker = 0.2. MED approach is utilized as the aggregating algorithm, and assuming full replenishment is used. That means, same number of proper workers are required to be added as the number of removed spammers. The experimental results are shown in Table 3.5. The "Optimal Order" represents the process of spam removing and de-biasing as: Eliminating random and uniform spammers + De-biasing + Removing sloppy workers.

Although with the order of "Spam Filtering + De-biasing", many biased workers are removed as sloppy workers (spammer), there is no significant difference of the results compared to the "Optimal Order". This is because all the removed workers are replaced with new proper workers, and it explains why there are much more new proper workers added in "Spam Filtering + De-biasing" than "Optimal Order". More workers mean more cost would be spent, which is not desired.

Comparing the results of "De-biasing + Spam Filtering" and "Optimal Order", the calculated metrics for former order is significantly worse than the latter. This is because the special case is taken into account in the experiments: let the true labels of a subset of items be $\{1, 1, 1, 1, 4\}$. Assume a worker w provides labels for the same subset of items as $\{5, 5, 5, 5, 5\}$. With the order of de-biasing first, w is detected as biased worker and his labels are then de-biased. Thus in the spam filtering period, w will not be rejected as a uniform spammer. Although the labels provided by w got de-biased, they will still degrade the final results when compared to the answers from proper workers. From Table 3.5, it could be seen that fewer new workers are added for "De-biasing + Spam Filtering" order in contrast to "Optimal Order", however, this is not always true. It is possible that a spammer s be detected as biased worker and his labels get biased. But the de-biased labels might still not good enough and s will be rejected in spam filtering period.

(III) Experimental results of removing spammers and de-biasing

The SRDF framework works the following way after getting the labels from crowds:

(1) Apply spam detection algorithm: After getting required number of labels received for each item, the spam detection algorithm will be applied to filter all the spammers. The detected spammers are rejected, as well as the labels they provided. Before detecting sloppy workers, de-biasing algorithm is required to be applied to remove biases from the answers from biased labelers.

(2) Add new workers: New workers are then added to give labels to replenish the removed spammers. If the new workers added are still spammers, keep on adding until enough proper workers. Instead of adding the same number of proper workers as the removed spammers, different amount of new proper labelers are examined to show its influence on the final results.

(3) Apply pattern detection approach: Bias patterns are detected for the workers by using the proposed detecting algorithm. Biases are then removed from the biased workers$'$ labels.

(4) Get consensus results: Finally, aggregation approaches such as AVG and Vancouver algorithm are applied to get the consensus labels for each item.

In order to test the improvement of results after applying SRDF framework, we run simulation with the percentage of random spammers as 10%, uniform spammers as 10% and sloppy workers as 10%. More combinations of different spammers population will be discussed later, we only consider 10% for each type here. From the outcome of experimental heuristics, it is found that the best threshold for both RandomSep and UniformSep are 1.0. Workers with $RandomSep$ higher than 1.0 or $UniformSep$ greater than 1.0 are removed. The accuracy chosen to filter sloppy workers in this study is 0.5, which determines all the labelers whose accuracy on categorical labels less than 0.5 are sloppy workers. After removing the spammers, we replenish the same number of proper workers as the removed spammers. The biased workers was originally set to 20% among the proper labelers. In our experiment, it is shown that by correcting biased labels through median of $diff(t_i, o_i)$, which is calculated from equation (3.1), gives optimal performance. Results are shown in Table 3.6.

In Table 3.6, the "Spam" rows denoted the metrics calculated without removing the spammers and de-biasing, "Spam Filter" rows are the outcome obtained after removing the spammers, and "SRDF" refers to the results obtained by applying SRDF approach. By comparing "Spam" and "SRDF" rows, we could see that there is significant improvement for σ, ρ and $RMSE$ after removing spammers, no matter how many workers used for each item and which aggregating approach is used. For instance, when each item is evaluated by 4 workers, by applying SRDF, we are able to reduce

TABLE 3.6

Results of evaluation metrics for AVG, MED, and Vancouver algorithm before & after applying spam filtering and SRDF framework (N_w: Number of workers for each item, Random Spam=10%, Uniform Spam=10%, Sloppy Worker=10%, Bias=20%)

Metrics			σ	ρ	$RMSE$
$N_w = 4$	AVG	Spam	6.13	0.77	6.86
		Spam Filter	3.79	0.94	3.50
		SRDF	2.80	0.96	2.93
	MED	Spam	6.05	0.79	6.73
		Spam Filter	3.80	0.94	3.78
		SRDF	**2.69**	**0.96**	**2.69**
	Vancouver	Spam	5.63	0.83	5.82
		Spam Filter	3.97	0.92	4.20
		SRDF	3.09	0.95	3.23
$N_w = 6$	AVG	Spam	4.96	0.85	6.02
		Spam Filter	2.80	0.96	3.04
		SRDF	2.41	0.97	2.60
	MED	Spam	4.38	0.88	5.09
		Spam Filter	3.04	0.95	3.68
		SRDF	**2.01**	**0.97**	**2.15**
	Vancouver	Spam	3.98	0.90	4.90
		Spam Filter	3.12	0.95	3.77
		SRDF	2.91	0.96	3.20
$N_w = 10$	AVG	Spam	4.13	0.91	4.79
		Spam Filter	2.10	0.97	2.96
		SRDF	1.88	0.98	2.43
	MED	Spam	3.69	0.94	3.94
		Spam Filter	2.36	0.96	3.66
		SRDF	**1.77**	**0.98**	**1.95**
	Vancouver	Spam	2.55	0.96	3.60
		Spam Filter	2.23	0.97	3.01
		SRDF	1.96	0.97	2.89

the error of σ by 56% from 6.05 to 2.69, and enhance ρ up to 22%. The $RMSE$ error is lowered from 6.73 to 2.69, which is 60% improvement, in MED model.

Before filtering the spammers and de-biasing the labels, best results could be obtained by using Vancouver algorithm. That means, Vancouver shows better performance than baseline models. However, after applying SRDF, MED and AVG aggregating algorithms showed better performance than Vancouver. The best accuracy results for each N_w setting are highlighted in Table 3.6. The model which utilized SRDF method and applied MED as aggregating algorithm showed the best quality based on the selected metrics. It is also interesting that after filtering the spammers, AVG presented better performance than Vancouver algorithm.

To get better understanding of why better results obtained through baseline model compared to Vancouver, after removing spammers, we try to run simulations on combination of different percentages of different types of spammers. In addition, it also allows us to check how the estimated

TABLE 3.7

Results for combination of different percentage of spammers (N_w: Number of workers for each item)

Proportion of Spammers	Aggregating Alg.	Spam Alg.	σ	ρ	$RMSE$
Rand=0.1 Uniform=0.1 Sloppy=0.1 ($N_w = 6$)	AVG	Spam	4.96	0.85	6.02
		Spam Filter	**2.80**	**0.96**	**3.04**
	MED	Spam	4.38	0.88	5.09
		Spam Filter	3.04	0.95	3.68
	Vancouver	Spam	3.98	0.90	4.90
		Spam Filter	3.12	0.95	3.77
Rand=0.4 Uniform=0.3 Sloppy=0.2 ($N_w = 6$)	AVG	Spam	8.06	0.46	8.16
		Spam Filter	**2.98**	**0.95**	**3.19**
	MED	Spam	7.98	0.47	8.07
		Spam Filter	3.13	0.94	3.97
	Vancouver	Spam	7.63	0.59	7.86
		Spam Filter	3.51	0.93	4.11
Rand=0.3 Uniform=0.2 Sloppy=0.4 ($N_w = 6$)	AVG	Spam	6.72	0.66	6.73
		Spam Filter	**2.43**	**0.97**	**2.94**
	MED	Spam	7.06	0.58	7.16
		Spam Filter	3.02	0.95	3.53
	vancouver	Spam	5.47	0.73	5.94
		Spam Filter	2.60	0.96	3.27

true labels were affected by spammers. The number of workers for each item is fixed to set as 6. Table 3.7 presents the results. The reported metrics showed that with increasing of spammers proportions, less accurate results are given without spam removing. For instance, when setting Rand = 0.1, Uniform=0.1, and Sloppy=0.1, AVG, MED and Vancouver algorithms obtained σ as 4.96, 4.38, and 3.98 before spam removing. After increasing Rand to 0.4, Uniform to 0.3 and Sloppy to 0.2, σ now is 8.06 (> 4.96), 7.98 (> 4.38), and 7.63 (> 3.98) for AVG, MED, and Vancouver respectively. Another fact could be seen from Table 3.7 is that AVG always gave the best accuracy among the three models after removing spammers, regardless the proportion of different spammers. With 30%, 20%, and 40% for random spammers, uniform spammers and sloppy workers separately, AVG is able to obtain σ as 2.43 compared to 3.02 for MED and 2.60 for Vancouver. It confirms the results in Table 3.6 that AVG has better performance than Vancouver after removing spammers. AVG and MED performs worse than Vancouver before spam filtering is since Vancouver benefits from weighting workers differently according to their variances. However, after removing the spammers, Vancouver can no longer get any benefits from offering weights for proper workers, whom with consistent behaviors. Thus AVG gives better performance after spam filtering. As we discussed before, MED presented best results with SRDF utilized compared to AVG and Vancouver. Although MED has the best performance, the results of AVG are very close to those best calculated metrics.

TABLE 3.8

Results of metrics with increased percentage of biased workers

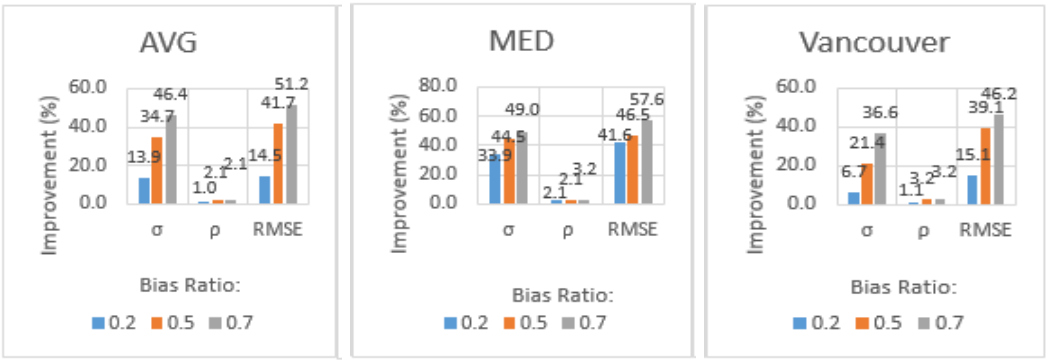
Rand=0.1, Uniform=0.1, Sloppy=0.1, $N_w = 6$			σ	ρ	$RMSE$
Bias Proportion=0.2	AVG	Biased	2.80	0.96	3.04
		De-bias	2.41	0.97	2.60
	MED	Biased	3.04	0.95	3.68
		De-bias	**2.01**	**0.97**	**2.15**
	Vancouver	Biased	3.12	0.95	3.77
		De-bias	2.91	0.96	3.20
Bias Proportion=0.5	AVG	Biased	2.85	0.95	3.38
		De-bias	1.86	0.97	1.97
	MED	Biased	2.92	0.96	3.61
		De-bias	**1.62**	**0.98**	**1.93**
	Vancouver	Biased	3.22	0.94	3.94
		De-bias	2.53	0.97	2.40
Bias Proportion=0.7	AVG	Biased	2.93	0.96	3.63
		De-bias	1.57	0.98	1.77
	MED	Biased	3.06	0.95	3.80
		De-bias	**1.56**	**0.98**	**1.61**
	Vancouver	Biased	3.39	0.94	3.98
		De-bias	2.15	0.97	2.14

To test the improvement of results after de-biasing the labels provide by biased workers, we increase the proportion of biased labelers. Table 3.8 gives the results while raising the ratio of biased workers. Three different bias percentages are considered: 20%, 50% and 70%. The "Biased" rows represent the results by only applying spam removal, without de-biasing, and the "De-bias" rows show outcome with removing spammers and de-biasing.

In Table 3.8, the models with more biased workers show better results after de-biasing, such as the best standard deviation obtained while $biasproportion = 0.5$ is 1.62, whereas 1.56 could be achieved when bias proportion increased to 0.7. The reason is that more biased workers means more predictable behavior patterns among the remaining proper workers. By correcting the labels from biased workers, better accuracy could be obtained. Figure 3.12 gives the detailed information of improvement (%) after de-biasing for each setting of bias ratio. The AVG approach has 13.9% improvement for σ, with bias ratio 0.2. When we increase the ratio to 0.5, the improvement can be achieved up to 34.7%. Similarly, better improvement is given for higher ratio of biased graders for both MED and Vancouver algorithm.

(IV) Experimental results of replacing spammers with different number of new workers

Adding new proper workers as a replacement for the removed spammers did give better



(a) Improvement for model with AVG algorithm

(b) Improvement for model with MED algorithm

(c) Improvement for model with Vancouver algorithm

Figure 3.12: Improvement results for different proportion of biased workers (Rand = 0.1, Uniform = 0.1, Sloppy = 0.1, $N_w = 6$)

performance comparing to results before spam removal, as shown before in Table 3.6. Considering most of the labelers remaining are proper workers after removing spammers, it is doubtful that same number of workers are still needed in contrast to the amount of workers before spam filtering. In many cases, budget for labeling tasks are limited, and the challenge becomes get labels with acceptable accuracy by the lowest cost. Thus, it is necessary to analyze the possibility of adding fewer new workers to reduce the cost for labeling, but maintaining the quality of consensus results.

In order to test the accuracy of results while adding different number of proper workers after removing the spammers, we run simulations with the prerequisite that there is a threshold for the number of workers for each item. This approach partially replenish the workers after removing spammers. We denote the threshold as N_{thr}, which represents that every item should be evaluated by N_{thr} workers. After rejecting the labels from spammers, each item will be checked and new proper workers are going to be added to ensure N_{thr} labels are received for the item. The process of worker replenishment in this case are discussed in the section of optimizing number of workers of the thesis. We set the N_{thr} as 3, and give the experimental results in Table 3.9. The rows indicated with "Full" are metrics calculated after adding same number of new proper workers as the removed spammers. The best results are obtained while maintaining same number of workers as initial setting after removing spammers. In Table 3.9, the highlighted results which with N_{thr} as threshold for submissions, however, do not have significant difference compared to the best results we could achieve. Figure 3.13 shows the average number of new proper workers added after spam removing for Table 3.9. N_w is the number of workers for each item initially. Less new workers are required to be added for keeping N_{thr} approach, and the accuracy remains acceptable. For example,

TABLE 3.9

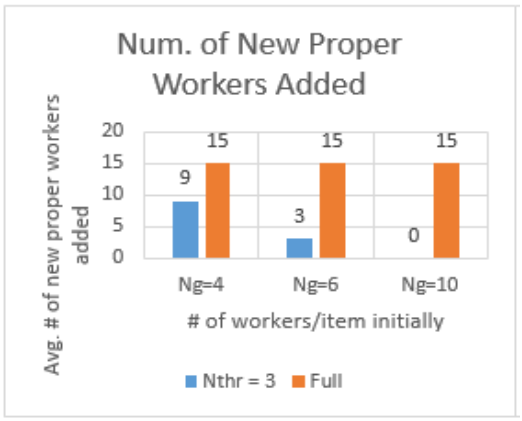
Results of different number of new proper workers added after spam filtering (N_w: Number of workers per item initially. Spam: No spam filtering)

					σ	ρ	$RMSE$
Rand=0.1 Uniform=0.1 Sloppy=0.1 Bias=0.2 (Workers population with small number of spammers)	$N_w = 4$	MED	Spam		4.89	0.83	5.47
			SRDF	$N_{thr} = 3$	**2.73**	**0.95**	**3.14**
				Full	2.68	0.95	2.95
		Vancouver	Spam		4.82	0.89	4.89
			SRDF	$N_{thr} = 3$	**3.59**	**0.92**	**3.67**
				Full	3.27	0.93	3.40
	$N_w = 6$	MED	Spam		3.72	0.90	4.17
			SRDF	$N_{thr} = 3$	**2.30**	**0.96**	**2.72**
				Full	2.12	0.97	2.28
		Vancouver	Spam		3.09	0.94	3.60
			SRDF	$N_{thr} = 3$	**2.56**	**0.95**	**2.91**
				Full	2.47	0.96	2.83
	$N_w = 10$	MED	Spam		3.45	0.93	4.09
			SRDF	$N_{thr} = 3$	**2.24**	**0.97**	**2.38**
				Full	1.81	0.98	2.01
		Vancouver	Spam		2.48	0.96	3.22
			SRDF	$N_{thr} = 3$	**2.13**	**0.97**	**2.25**
				Full	2.06	0.97	2.19
Rand=0.4 Uniform=0.3 Sloppy=0.2 Bias=0.2 (Workers population with large number of spammers)	$N_w = 4$	MED	Spam		10.60	0.25	10.71
			SRDF	$N_{thr} = 3$	**2.52**	**0.95**	**2.70**
				Full	2.29	0.96	2.38
		Vancouver	Spam		9.81	0.28	10.04
			SRDF	$N_{thr} = 3$	**3.62**	**0.92**	**3.73**
				Full	3.51	0.92	3.69
	$N_w = 6$	MED	Spam		9.46	0.40	9.55
			SRDF	$N_{thr} = 3$	**2.19**	**0.97**	**2.27**
				Full	2.11	0.97	2.18
		Vancouver	Spam		8.05	0.50	8.19
			SRDF	$N_{thr} = 3$	**3.02**	**0.94**	**3.58**
				Full	2.99	0.95	3.32
	$N_w = 10$	MED	Spam		9.16	0.42	9.37
			SRDF	$N_{thr} = 3$	**2.37**	**0.96**	**2.62**
				Full	1.64	0.98	1.69
		Vancouver	Spam		7.21	0.63	7.43
			SRDF	$N_{thr} = 3$	**2.78**	**0.95**	**3.18**
				Full	2.23	0.97	2.74

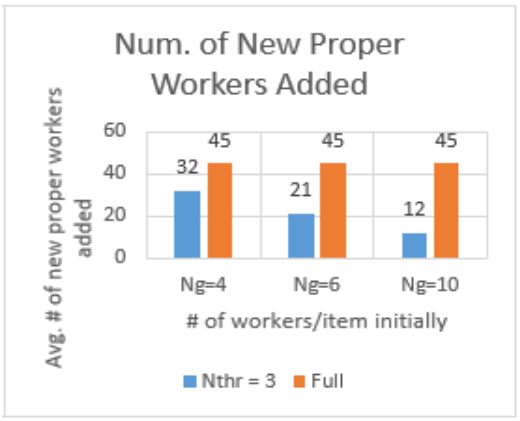
when setting Rand=0.1, Uniform=0.1, and Sloppy=0.1, no matter how many labelers evaluated each item initially, 15 new proper workers are required to replace the spammers for full replenishment. However, only 9, 3, and 0 new proper workers are added if we set N_{thr}=3 for initial number of labelers per item as 4, 6 and 10 respectively.

(V) Evaluation of proposed framework

We compared the SRDF framework with several other methodologies. First of all, the proposed framework in this proposal is compared with the work of Vuurens et al. [60], Alfaro and Shavlovsky [7], and Piech et al. [5]. The comparisons are shown in Table 3.10. We experimented with



(a) Rand = 0.1, Uniform = 0.1, Sloppy = 0.1



(b) Rand = 0.4, Uniform = 0.3, Sloppy = 0.2

Figure 3.13: Number of new proper workers added after spam filtering (Bias proportion=0.2)

TABLE 3.10

Percentage of reduction on RMSE for different methodologies

	BASELINE	METHODOLOGY	RMSE REDU. (%)
VURRENS ET AL. [60]	MED	Spam Removing w/MED	28%
ALFARO & SHAVLOVSKY [7]	MED	Vancouver	50%
PIECH ET AL. [5]	MED	Tuned Models	33%
PROPOSED FRAMEWORK	**MED**	**SRDF w/MED**	**58%**

all of the methodologies including MED baseline model with the synthetic data set, and reported the percentage of reduction on RMS error. It could be seen that by utilizing SRDF, we are able to reduce the RMSE by 58%, which is the best results among all the other compared approaches.

In addition, by applying SRDF to remove the spammers and de-bias the labels, we are able to employ less workers to achieve close enough accuracy comparing to other methods. For example, as shown in Table 3.9, by making sure each item reviewed by 3 proper workers, we are able to get even better results than using Vancouver algorithm with 6 workers per item.

We also compare SRDF framework with the algorithm proposed by Wang et al. [81], which separate true error rates from the biases. The average classification error in [81] was around 0.12 depends on the percent of known examples in gold data set. In our proposal, we could obtain 0.11 error rate by applying the proposed methodology.

3.7 Chapter summary

This chapter presents the influences of the spam workers and biased workers on the consensus results in crowdsourcing systems. A worker filtering method from existing literature is tuned and

adopted to remove the spam workers, for the purpose of improving the quality of estimated true labels. To address the biasing problem, we proposed an iterative bias detection algorithm to recognize the biased workers and correct the labels received from them. An integrated approach is then presented to combine both spam removing and bias detection algorithms to obtain high quality consensus results from the crowdsourced data.

To test the proposed SRDF (Spam Removing and De-biasing Framework), simulated data sets are generated for experimental analysis. By comparing to a set of baseline models, such as AVG and MED, it was found that by setting 1 as threshold for $RandomSep$ and $UniformSep$, and 0.5 as least required accuracy for sloppy workers to remove the spammers, significant improvement can be achieved on the consensus results. De-biasing the observed labels before aggregating them gives further 4% to 30% improvement on the evaluation metrics.

In addition, this chapter investigate the possibility of reducing the number of workers by utilizing SRDF. Experimentation on synthetic datasets was performed, and it was found that by setting a threshold on the number of labels received on each item, fewer workers, which lead to lower cost, can be employed while acceptable quality of estimated labels can be obtained.

CHAPTER 4

SLOPPINESS MITIGATION – DETECTING AND CORRECTING WORKER BIAS

In Chapter 3, a categorization of workers, including random spammer, uniform spammer, sloppy spammer, and proper worker, has been presented based on the existing literature [60]. Based on the worker groups′ behavior characteristics, worker filtering, which removes spammers and detects biased workers, is performed. However, the definition of the worker groups in Chapter 3 actually overlapped with each other among the crowd, such as random spammers and uniform spammers. As an example, if a worker w provides labels as $\{5, 5, 5, 5, 5, 5\}$, for a set of item with truths as $\{2, 2, 2, 2, 2, 2\}$. The calculated $RandomSep$ is greater than 1, which indicates w as a random spammer. At the same time, the estimated $UniformSep > 1$, which denotes that w also belongs to uniform spammer group. As indicated in [60], random spammer "gives useless answers without reading the question or data", whereas uniform spammer "chooses one label to primarily vote, sometimes switching labels on different types of questions". Based on this definition, it can be seen that uniform spammer also gives useless labels without making any efforts for the task, which is very similar to random spammers. Thus, it is not necessary to distinguish them separately. As such, a different way of classifying workers is required in our work to investigate the worker behaviors in crowdsourcing systems. In addition, the sloppy workers are removed as spammers in Chapter 3, and removal of this type of workers has great impact on the detection of biased workers. In this chapter, we overcome the dependence between removal of sloppy worker and detection of biased worker, by classifying the biased workers as a sub-category of sloppy worker group. Instead of removing the sloppy workers, the work in this chapter utilizes the labels provided by them. A probabilistic based bias detection model is developed to examine whether a worker is biased or not. Finally, an optimization based framework is utilized to derive true labels from the observed labels from crowd. Our research in this chapter concentrates on the crowd scoring tasks with ordinal labels.

4.1 Sloppiness in crowdsourcing systems

The scale and diversity data from various sources leads to information explosion and challenge of big data in recent years. These data are generated with wide variety, i.e., social networks such as Twitter, Facebook or Linkedin; business or entertainment platforms, such as Amazon, IMDb, or Netflix; and many other Internet resources, such as the Massive Open Online Classes (MOOCs). Many of the data processing tasks require the involving of human intelligence, such as image labeling, video annotation, natural language processing, machine translation and product recommendation [7,105,118,119]. This stimulates the emerge of crowdsourcing systems, which is human powered problem-solving methodology for collecting labeled data. Crowdsourcing has been demonstrated as a effective and important approach among others by Amazon Mechanical Turk, reCAPTCHA [25], Duolingo, and the ESP Game, etc [46, 112]. Different from traditional way of solving problems through experts, crowdsourcing mitigates the expensiveness of time and financial cost for large scale tasks. However, despite the promises, key challenge within the crowdsourcing systems exists: quality of the collected data is unknown, and is highly noisy in many cases. This is due to the fact of wide ranging expertise levels of workers and difficulty levels of tasks. To obtain high quality labels or answers for the crowdsourcing tasks, it is crucial to identify the trustworthiness of the workers.

The trustworthiness defines the reliability of a worker. In order to aggregate different answers provided by crowd workers, it is intrinsic to concentrate more on the reliable workers instead of untrustworthy ones. The factors which influence the reliability of a worker include expertise level, personal preference, understanding of the tasks, and worker's interests [46]. Due to the tedious and low reward nature of crowdsourcing tasks, errors are common even for workers who make an effort [105]. Ipeirotis and Gabrilovich [134] mentioned in their research that the monetary incentive, which is the case in many crowdsourcing platforms, is a mixed blessing: It might attract workers, but has the probability to make things worse [121,122]. Similar to truth discovery tasks [8,123–126], it is desired to discover true answers and worker reliabilities from multiple crowd answers. Table 4.1 presents the definitions of truth discovery tasks and crowdsourcing aggregation tasks, their similarities and differences [8].

A lot of research has been proposed to investigate the problems of inferring true labels and worker reliabilities in crowdsourcing for binary labeling tasks [41, 50, 71, 127]. The real world applications, however, are not always with just yes or no answers. There are many sophisticated labeling tasks with more than two choices available. Here, we focus on ordinal crowd labeling tasks,

TABLE 4.1: Truth Discovery & Crowdsourcing Aggregation: definitions, similarities, and differences [8]

	Truth Discovery	Crowdsourcing Aggregation
Definition	Integrates multi-source noisy information by estimating the reliability of each source.	Aggregate noisy answers contributed by crowd workers to obtain the correct answers.
Similarities	1. Both are trying to find trustworthy and accurate information from multiple sources.	
	2. Their goals are to improve the quality of aggregation results.	
	3. They have similar principles: reliable sources(workers) tend to provide high quality information; sources are reliable if they provide accurate information.	
	4. Techniques used are similar, and ground truth is usually unavailable in both cases.	
Differences	Passive: data is already generated, it is available when we find it.	Active: user is able to choose what and how much data to generate.
	Data crawled from online Web or collected from databases might have various types and may change dynamically.	More information might be accessed on the source features, such as workers' location, accuracy on historical tasks, and education background, etc.

which is also called crowd scoring tasks, with ordinal and multiple categories of labels. For example, grading project submissions from students in a class could be considered as a labeling task. Assume the scale of the labels is from 1 to 5. In this case, 4 is closer to 5 than 2. That means, giving label 4 to a submission, whose true label is 5, is different from providing label as 2. To obtain the true labels and worker reliability, a vast variety of techniques have been proposed on the basis of principle that reliable workers tend to provide true labels, and truth should be reported by many reliable workers [128]. Most of the existing approaches measures worker reliabilities according to their accuracies (e.g. inverse of variance [7]). The labels from more reliable workers contribute more to truth computation. However, these types of methodologies ignore one group of workers: highly biased and hence inaccurate workers. Biased workers are referred to those who constantly provide higher (or lower) label values compared with true labels. In addition, the labels obtained from the biased workers have the some patterns which could be used to extract useful information. As an example, assume the true labels for a set of items which need to be labeled are $\{3, 3, 3, 3, 3, 3\}$. We obtained labels for this set of items from two different workers. $w1$ as one of them provides corresponding labels as $\{3, 3, 3, 3, 3, 2\}$. $w2$ as the other one offers $\{2, 2, 2, 2, 2, 3\}$. Although $w1$ provides higher quality labels in this case, $w2$ actually offers equal amount of information as $w1$. In this case, the pattern shown in labels from $w1$ is 1 scale higher than true labels in most of the observed data. By correcting the labels provided by $w2$ through adding 1, the accuracy of $w2$ equals to $w1$. As what Passonneau and Carpenter [129] proposed in their work: A biased and hence inaccurate annotator can provide as much information as a more accurate annotator.

This type of biased workers belongs to the crowd group which we call workers with sloppy behaviors. The term sloppy is based on the work of [60], which describes a worker who "views the question and data, but maybe insufficiently precise in their judgments" as sloppy worker. The similar definitions could also be found in [130]. Sloppiness is the phenomena of observed labels fluctuating

around the true labels. The fluctuation could be caused by personal preference, misleading task description, or just fatigue after long time working, etc. Different from the spam labels, which are labels independent from truths and provide no useful information [50,129], the labels with sloppiness could still be utilized in the process of inferring the truth. A lot of researches have been done to recognize or filter the spam/useless labels provided by crowd workers. As an example, the answers which are randomly generated by workers are one type of spam labels. In contrast, we found few discussions about the sloppiness within the crowd. Our goal is to identify the workers with sloppy behaviors, especially the biased workers, and extract useful information from biased labels in order to get estimated labels as close as possible to the true labels. The true labels, which are also called gold truths, are defined as the labels provided by experts in our research. It is, however, not reasonable to always have the gold truths available. For example, in a relevance judgment task, which requires to rate the relevance of (query, URL) pairs, there might be millions of pairs that need to be rated. It would be too expensive and time consuming to hire experts to sit down and do all the judgments in order to get the gold truths. With the absence of the gold truths, it is challenging to distinguish different behaviors of workers, and thus difficult to infer truths from the observed labels. For example, when the true grade is known for a student submission in a class is A, on a scale of $\{A, B, C, D, F\}$. It is easy to justify a grader is "reliable" or not by just comparing the observed grade with the true grade. A reliable worker is defined as "Performs the tasks as requested. Reads the question and data and judges sufficiently precise" [60]. However, without knowing the true grade (in this case, assume true grade A is unknown), we could not simply claim whether a worker provide high quality grades or not.

In this chapter, we present the proposed iterative self-correcting truth discovery algorithm, to deal with the problem of unavailability of true labels, as well as making use of the labels provided by crowd workers with biased behavior. This approach is able to:

(a) Effectively identify the biased workers: A bias score is calculated for each worker, in order to determine whether he/she belongs to biased worker group.

(b) Correct the labels obtained from biased workers: According to the identified bias pattern, we de-bias the observed labels. Bias pattern is recognized as the behavioral feature of the workers. For example, the positive (or negative) bias pattern discussed in our work indicates the feature of worker providing labels constantly higher (or constantly lower for negative pattern) than the truths.

(c) Utilize the truth discovery framework to iteratively update the truths and worker reliabilities. Here, the computed worker reliability is obtained after removing the bias, which reflects the actual information a worker could provide. The reliabilities are utilized to weight the labels provided by each worker to obtain estimated true labels. More reliable workers are assigned with higher weight.

The chapter is organized as follows: Section 4.2 presents the related work on discovering truth on noisy data, and background research about dealing with worker biases. Section 4.3 presents the Iterative Self Correcting - Truth Discovery (ITSC-TD) algorithm, which use probabilistic way to recognize highly biased workers, and estimate true labels from the crowdsourced noisy data. Experimental results on both synthetic and real world data sets are presented in Section 4.4. Conclusion and chapter summary is presented in Section 4.5.

4.2 Related work on truth discovery and worker bias analysis

In Chapter 2, a brief review on aggregating algorithms to obtain consensus labels via repeated labeling has been presented. Two different types of approaches are introduced: Iterative, and Non-iterative methods to aggregate the labels from the crowd. Here, we present a different taxonomy on aggregating approaches with more state-of-art algorithms:

(1) Non-iterative methods.

(2) Iterative methods.

(3) Probabilistic Graphical Model (PGM).

(4) Optimization based truth discovery model.

4.2.1 Background work on aggregating algorithms

Majority Voting (MV) as the simplest aggregating method, is a non-iterative algorithm while achieving the consensus results. MV assumes high quality workers are the majority among the crowds and they work independently from each other. It assigns the same weight to all the workers, and then update the truths. Other non-iterative methods, such as average or HoneyPot (HP) [66], simply use heuristics to compute consensus values for items. The remaining three categories of aggregating algorithms are presented as follows: Section 4.2.1.1 presents the review on the iterative

aggregating algorithms and the PGM methods. The optimization based truth discovery models to obtain aggregated values for the items, which need to be labeled, are presented in Section 4.2.1.2.

4.2.1.1 Review of iterative and Probabilistic Graphical Model (PGM) aggregating algorithms

Relevant work on aggregating crowd labels has been reported by other research [45, 46, 71, 121, 123, 131]. ZenCrowd [121] was proposed by Demartini, Difallah, and Cudré-Mauroux. It was an extension of MV, which weights the workers' answers by their corresponding reliability. The approach uses Expectation Maximization (EM) to simultaneously estimate the true labels and worker reliability. Dawid and Skene's [46] approach models a confusion matrix for each worker, as well as the class prior. They proposed to use EM to estimate true labels, confusion matrix, and the prior in their work. Snow et al. [42] utilizes a similar model for human linguistic annotation. They consider the fully supervised case of machine learning estimation. Whitehill et al. [45] proposed GLAD, which stands for Generative model of Labels, Abilities, and Difficulties, to simultaneously infer the expertise of each worker, the difficulty of each item, and the most probable label of each item. Raykar et al. [71] use a Bayesian approach to add worker specific priors for each class. Their algorithm evaluates the different experts and gives an estimate of the actual hidden labels by using an automatic classifier. Raykar's approach requires the feature representation of the items, however, which is not always available. If such feature representation does not exist, the method falls back to maximum-a-posteriori (MAP) estimation. Zhou et al. [131] utilize a minimax entropy principle to estimate the true labels from the crowd answers. Their method assumes that labels are generated by a probability distribution over worker, items, and labels. By minimizing the Kullback-Leibler (KL) divergence between the probability and unknown truth, they infer the item confusability and worker expertise. Zhou's method is a natural extension to Dawid and Skene's work [46], and the essential difference is that the minimax entropy takes into account item confusability, in addition to worker expertise. Ertekin, Hirsh, and Rudin [123] present an algorithm called "CrowdSense" that works in an online fashion to dynamically sample subsets of workers based on exploration/exploitation criterion. The algorithm produces a weighted combination of the subset of workers' votes to approximate the crowd's opinion.

All the work mentioned above belongs to either iterative methods or probabilistic graphical model (PGM) based methods to infer the true labels."CrowdSense" is an iterative method, and the remaining presented algorithms are PGM based methods. Li et al. [18] provided a survey

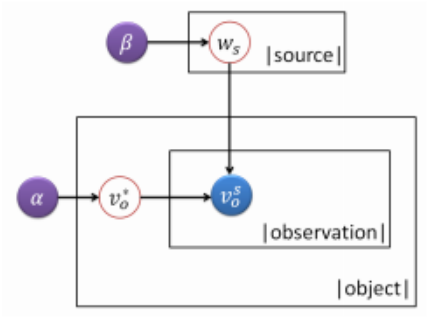


Figure 4.1: The general probabilistic graphical (PGM) model. [18]

on these methods in their work. Here is a summary of the iterative aggregating algorithms: in iterative methods, truth computation step and weight estimation step are iteratively conducted until convergence. In truth computation step, truths are inferred, while worker weights are assumed to be fixed. In weight estimation step, the workers' weights are updated based on the current estimated truths. PGM models, however, incorporate the principle of truth discovery: If a worker provides trustworthy labels frequently, he will be assigned a high reliability; if a label is provided by a reliable worker, it will have higher probability to be chosen as truth. The corresponding likelihood of a PGM model is presented in formula (4.1).

$$\prod_{s \in S} p(\omega_s|\beta) \prod_{o \in \emptyset} (p(v_o^*|\alpha) \prod_{s \in S} p(v_o^s|v_o^*, \omega_s)) \tag{4.1}$$

where v_o^s is the labels provided by the worker s for item o, v_o^* is the true label for the item, and ω_s is the worker$'$s weight. α and β are the hyper-parameters correlated to the truth and worker reliability. The graphical representation of the general model is shown in Figure 4.1. To infer the hidden true labels and worker weights, techniques such as Expectation Maximization (EM) and Gibbs Sampling could be adopted.

4.2.1.2 Review of optimization based truth discovery models

In addition to the inference and PGM models, there is another type of algorithm that could be utilized to generate truths from crowd answers – optimization based truth discovery methods [119, 128, 132]. This type of approach captures the true labels by solving the optimization problem, which is in the following formulation:

$$\arg\min_{\{\omega_s\},\{v_o^*\}} \sum_{s\in S} \omega_s \sum_{o\in\emptyset} d(v_o^s, v_o^*) \tag{4.2}$$

where $d(\cdot)$ is the loss/distance function between the crowd answer and identified truth. By minimizing Eq. (4.2), the aggregated results (v_o^*) will be closer to answers from workers with higher weights. Meng et al. [128] proposed an effective optimization based truth discovery framework to infer the truths for crowd sensing of correlated entities. Aydin et al. [119] investigate a novel weighted aggregation method to improve accuracy of crowdsourced answers for multiple-choice questions. They deploy the optimization based truth discovery algorithm, as well as the light weight machine leaning (ML) techniques for building more accurate crowdsourced question answering systems. Li et al. [132] propose to identify the true information among multiple sources of data by using an optimization framework. Their model treats the truths and source reliability as unknown variables. The objective is to minimize the overall weighted deviation between truths and observations. They also discussed different types of loss functions which could be incorporated into the framework.

As stated in [18]: there are differences between the three types of aggregation or truth discovery methodologies, however, we do not claim that one of them is better than another. It is, however, possible to see the advantages of different approaches in various cases. Li et al. [132] proved superiority of the optimization based methods in their work by comparing the results of the proposed framework with some Bayesian analysis based approaches, on heterogeneous data. In terms of interpretability, iterative methods are easier to understand than others. PGM models and optimization based approaches could take into account the prior knowledge about true labels and workers compared to iterative algorithms. We utilized the optimization based framework in this chapter since it is easier for us to correct the highly biased crowd workers in coordinate descent process. In addition, it will be easier to extend to multiple data type cases in the future, such as crowd answers that contain both numerical and categorical ordinal data.

4.2.2 Background work on worker bias analysis

Much of the literature has accounted for the labeler bias [31, 42, 47, 71, 72, 75, 120]. As Wauthier and Jordan [75] stated, the data collected from crowdsourcing services is often very noisy: Unhelpful labelers may provide wrong or biased answers which may greatly degrade the learning algorithms. Bias may be caused by personal preference, systematic misunderstanding of the labeling task, lack of interest, or malicious intents. When the levels of bias are low, some of the consensus

or aggregating algorithms could still work well, but could become unreliable when the quality of workers vary greatly [41]. For example, when extreme biased workers exist in a binary labeling task, the EM model proposed by Dawid and Skene [46] is able to flip the labels to achieve higher accuracy. However, if too many of the workers are highly biased, the model cannot separate the noise from the true labels [129].

Snow et al. [41] proposed a multinomial model similar to Naive Bayes for modeling labels and workers. They estimate the worker quality in a Bayesian setting by comparing with the gold standard, and apply the weighted voting rule which give highly biased workers negative votes. In this way, they correct the bias in categorical data. Wauthier and Jordan [75] presented Bayesian Bias Mitigation for Crowdsourcing (BBMC), a Bayesian model to capture the sources of bias by describing workers as influenced by shared random effects. These effects are also known as the latent features which depict the preferences of the workers. Raykar et al. [71] utilized a Bayesian approach to capture the mislabeling probabilities by assigning latent variables to workers. Ipeirotis et al. [31] proposed to separate error and bias for the workers. They pointed out that a biased worker is still more useful than a worker who reports labels at random. Dekel and Shamir [120] presented a two step process to pruning the low quality worker in a crowd. They first remove the workers by how far they disagree with the estimated true label, and then they reuse the cleaned data set to build the model. Yan et al. [47] employed a coin flip observation model to learn the worker bias, and then optimally selecting new training points and workers. Welinder et al. [72] modeled the worker in an image labeling process as a multidimensional entity with variables representing competence, expertise, and bias. Their work generalize and extends the research of [45] by introducing worker bias. The authors in [11] introduced and evaluated probabilistic models that can detect and correct task-dependent bias automatically. Zhang et al. [133] proposed an adaptive weighted majority voting (AWMV) algorithm to handle the issue of biased labeling. Their work is based on the statistical difference between the labeling qualities of the two classes.

Most of the research mentioned above assume that the task is binary labeling [31,71,75,133]. Although many of them claim it could be generalized to more than two labels, it is difficult and have high computational complexity. Some of the work tackles the bias workers, however, they didn't really separate the biased ones from noisy spam workers. The authors in [129] gives discussions about the features of the biased annotators. They, however, solely utilized expectation maximization (EM) based probabilistic model for a word sense annotation task as a case study. They do not handle the biased labeling in [129]. The model proposed by Snow et al. [42] requires the gold truths to be

compared to observed labels in order to recognize biased workers. In [31], the methodology would work only if each worker provide at least a specified number (20 to 30) of labels, which is difficult to fulfill in many cases. The BBMC model proposed by Wauthier and Jordan [75] captures the sources of labeler bias through shared random effects. Different from their work, we take into account the overall effect of the bias to obtain high quality estimated labels. In the work of [11], task-specific bias, such as confusing a specific class with another, is captured by utilizing the task features. Our work, however, accounts for the biases of workers through the observed labels. In other words, we do not rely on the features of the labeling task to account for biases. In [133], the authors adapted a majority voting algorithm to consider the bias of workers through bias rate in a binary labeling task. This chapter, unlike [133], proposed an approach to model biases on an optimization based truth discovery framework for ordinal labeling tasks. The authors in [72] modeled the label assigned by a worker according to a linear classifier. The classifier is parameterized by a direction and a bias parameter, and the model is developed under the assumption that the labels are binary. In contrast, our approach deals with the worker separation problems in a labeling task with multi-valued ordinal labels. Instead of the workers without information offered in the labels, we focus on correcting the biased workers to improve the quality of the estimated truths.

4.3 The Iterative Self Correcting – Truth Discovery (ITSC–TD) methodology

The proposed iterative self correcting – truth discovery framework recognizes the highly biased workers, and computes the truth and weights from the bias-corrected labelers. The model consists of two steps: (1) Bias Correcting: Compute a bias score for each crowd worker, and according to the score, determine whether the worker belongs to the highly biased group. If he/she is highly biased, de-bias the worker. (2) Truth Discovery [18,132]: Formulate the truth computation problem as an optimization problem which models the truths as weighted voting of the biased corrected labels from step 1. The model could be easily generalized to numerical labeling tasks. The problem formulation, which describes the problem we will solve in this chapter, is presented in Section 4.3.1. Section 4.3.2 presents the proposed bias detection model on sloppy workers. Section 4.3.3 presents the iterative self correcting truth discovery framework for estimating truths from noisy crowdsourced data.

4.3.1 Problem formulation

We give detailed description of the problem which we will solve in this chapter, and it then serves as the foundation of the proposed framework. Before giving a definition of the problem, we introduce the different types of workers in a labeling task to give a better understanding of the worker behaviors. Section 4.3.1.1 presents the background of the research, which gives a hierarchical categorization of the crowd in crowdsourcing systems. Section 4.3.1.2 presents the formalization of sloppiness.

4.3.1.1 Background

There are two general types of workers: reliable workers and unreliable workers [60,129,130]. Reliable workers are ones who would be able to provide high quality labels. The definition of this type of worker is similar to the concept of "proper worker" from [41]: the worker which "performs the tasks as requested. Reads the question and data and judges sufficiently precise". Unreliable workers, however, give labels which would have an uncontrolled effect, or even negative influences on obtaining truths, by utilizing learning/aggregating algorithms. We use the term "noisy worker" to represent unreliable workers, due to the fact that the labels provided by this type of worker are highly noisy. Based on the different behavior patterns of the noisy workers, many research has been proposed to categorize these workers. For example, Vurrens et al. [60] categorized noisy workers into random spammer, uniform spammer, and sloppy worker; Kazai et al. [130] defined a topology of noisy workers as sloppy workers, incompetent workers, and spammers; and Passonneau and Carpenter [129] proposed spam annotators, biased annotators, and adversarial annotators. Although different literature give different names or definitions, the categories of noisy workers could be generalized into two different groups:

(i) Spam workers: This type of worker provides useless labels, which means not so much information could be extracted and utilized for the aggregating process. Instead, they would greatly degrade the estimated true labels. For example, random spammers and uniform spammers in [60], spammers in [130], and spam annotators in [129] all belong to this group of worker.

(ii) Useful low quality workers (sloppy workers): This type of worker is very special. They provide low quality labels, but after processing the labels, we could obtain high quality results. For example, in a binary labeling task, flipping a worker′s provided labels, whose accuracy is 0.3, could resulting 0.7 accuracy on the flipped labels. The adversarial annotators, which is also

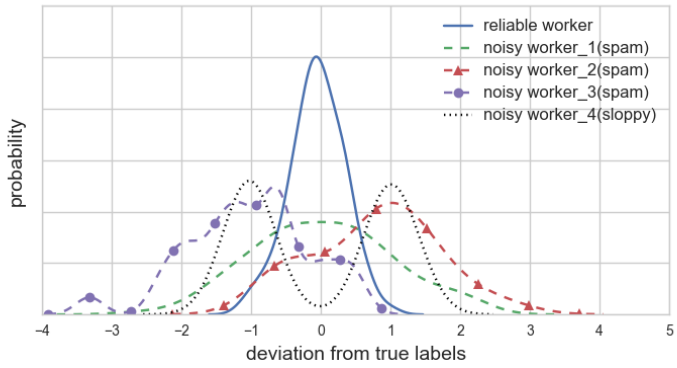


Figure 4.2: Illustration of reliable workers and noisy workers

extreme biased workers, mentioned in [129] falls into this group of worker. The sloppy workers and incompetent workers presented in [130] also belong to this category, according to their definitions. For easier reference, we call this type of worker as sloppy worker. Our work concentrates on dealing with sloppy workers in this chapter.

We give visual presentation between reliable workers and noisy workers in Figure 1.2. The x axis denotes the deviation between the observed labels from workers and the true labels, and the y axis is the probability of the deviation. In Figure 4.2, the reliable worker showed higher accuracy compared with noisy workers. A perfect worker would have probability as 1 at the point of $deviation = 0$. Four different examples of noisy workers were presented. Among them, "noisy worker_1", "noisy worker_2", and "noisy worker_3"are the spam workers, and they do not provide useful information in the labels. For example, the "noisy worker_1" has low accuracy, and errs on both sides of deviations. The distribution of the deviations spreads over the x-axis. While "noisy worker_2" mainly has random errors towards the right side (positive) of the deviations. The "noisy worker_3", however, tends to give random labels which lead to negative deviations. The "noisy worker_4"in the figure belongs to the sloppy worker group. Similar to spam workers, a sloppy worker has low accuracy. Unlike spam workers, a sloppy worker does not give random labels. In Figure 4.2, the sloppy worker provides labels close enough to truths.

It is much easier to distinguish between the spam workers and sloppy workers in a binary labeling task. This is due to the nature that the accuracy decision threshold of spam worker is fixed, which is 0.5, for binary labeling tasks [41]. As an example, if worker's accuracy is higher than 0.5, the worker could be classified as a reliable or good worker. If a worker's accuracy is less than 0.5, by flipping the labels he/she provided, we could still get acceptable results. Only when the probabilities

TABLE 4.2

Examples of confusion matrix of workers

Observed Labels		A	B	C	D	F
True Labels	A	**0.70**	0.30	0.00	0.00	0.00
	B	0.30	**0.60**	0.10	0.00	0.00
	C	0.00	0.00	**0.80**	0.20	0.00
	D	0.00	0.00	0.40	**0.50**	0.10
	F	0.00	0.00	0.00	0.30	**0.70**

(a) Diagonal of correct labels for reliable workers [46]

Observed Labels		A	B	C	D	F
True Labels	A	**0.40**	0.60	0.00	0.00	0.00
	B	0.10	**0.20**	0.70	0.00	0.00
	C	0.00	0.10	**0.30**	0.60	0.00
	D	0.00	0.00	0.00	**0.20**	0.80
	F	0.00	0.00	0.00	0.50	**0.50**

(b) Diagonal of correct labels for sloppy workers

of choosing one of the two labels are equal, the worker is considered as a spammer. In real world applications, the scale of the labels actually entails more than two levels. As an example, a student′s test grade is not always only fail or pass, the scale is usually like "A, B, C, D, F ". In this case, the threshold 0.5 is no longer applicable due to the fact that we cannot just simply flip the labels. To make use of the labels from sloppy workers, we need to come up with a different strategy.

Since the accuracy measure could not be used to separate sloppy workers from spam workers, we investigate one important attribute discovered by Dawid and Skene [46]. According to the research in [46], mostly, even though reliable workers make errors, their errors will be on the diagonal of the true labels. As an example, Table 4.2(a), (b) presents the confusion matrix of a reliable worker. Instead of giving the results as the number of instances, the matrix shows the ratio of different observed labels to true labels. The highlighted cells give accuracies based on truths, such as 0.7 in the cell with True Labels as 'A' and Observed Labels as 'A′ is calculated as:

$$\begin{aligned} Acc_A &= Pr(obsLabels = A | tLabels = A) \\ &= \frac{|es_A|}{|ts_A|} \end{aligned} \tag{4.3}$$

where "obsLabels" are the observed answers from workers, and "tLabels" represents the true labels. The ts_A is a set where all the tasks have true labels as 'A', and $es_A \subseteq ts_A$, which contains the tasks with observed labels as 'A'. It could also be interpreted as: if the task has true label as 'A', the worker has the possibility of 0.7 to provide the correct label ('A') to this task. Similarly, the cell(A, B) gives the possibility of the worker to provide grade 'B′ for a task with true grade as 'A'. As indicated in Table 4.2(a), the worker only err one scale up or down from the true labels. As an example, when the true label is 'B', although the worker might make error, he/she only shows possibility > 0 of labeling the task as 'A′ or 'C'. The possibility of observed grade as 'D′ or 'F′ is 0.

However, let us investigate another example of worker w's confusion matrix as shown in Table 4.2(b). Comparing the confusion matrix of reliable worker in Table 4.2(a), it could be seen that w in Table 4.2(b) provides lower quality labels. However, same as shown in 4.2(a), w also presented the diagonal of correct labels attribute. After combining the characteristics of different types of workers and the discovery, we give a detailed definition for sloppy workers in our work: sloppy workers have lower accuracy than reliable workers, at the same time, they are like reliable workers: they only err on the diagonal of the true labels. The "noisy worker_4" in Figure 4.2 showed the interesting fact that the deviations are chosen from $\{0, -1, +1\}$. "Err on the diagonal" in this example indicates the worker makes error either as $+1$ and -1. For different real world applications, the diagonal attribute might be as: errors/deviations can be selected from $[-a, +a]$ instead of just $+1$ or -1. For example, while grading a student$'$s homework from class, a reliable worker might make errors between $[-5, +5]$ on a grading scale from 0 to 100. We call the $[-a, +a]$ as the fault tolerance range. That means, if a worker only errs on this range, he/she did have the "diagonal attribute". The following example gives an intrinsic motivation of defining the range for sloppy workers: Assume a multivalued ordinal labeling task, and the possible label set is $\{1, 2, 3, 4, 5\}$. There are 10 items which need to be labeled, with truths as $[2, 3, 3, 4, 3, 2, 3, 5, 4, 3]$. Suppose two workers w_1 and w_2 worked on the task. The labels provided by w_1 as $[3, 4, 4, 5, 4, 3, 3, 5, 5, 4]$, and labels from w_2 are $[5, 5, 5, 5, 5, 5, 5, 5, 5, 5]$. Both of the workers in the example have low accuracies $w_1 = w_2 = 0.2$. In addition, both of them tend to have positive deviations for most of their labels compared to truths. The amount of information, however, provided by w_1 and w_2 is different. By correcting the labels of w_1 through subtracting 1, accuracy as 0.9 could be obtained. While w_2's labels offer no information regarding approximating the truths. From the example, we could see that by defining the fault tolerance range, it would be possible to utilize the labels from w_1, while removing w_2 as spam worker.

To determine the value for a, which is used in the fault tolerance rang, here are some guidelines:

(1) As discussed above, we constrain the deviation range of sloppy worker under acceptable range, which is similar to reliable workers. One way to choose a would be applying binary division heuristic approach to decide the value. Here is how binary division heuristic approach could be conducted. Assume the possible ordinal label set as L.

   (a) First set $a_0 = |L|/2$, where $|L|$ is the total number of possible ordinal labels. Calculate the estimated truths and selected performance metrics such as accuracy or F measure, by

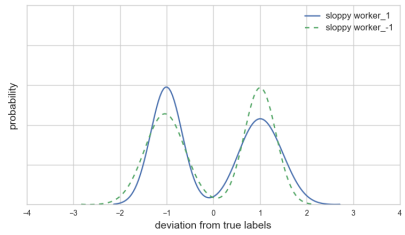


Figure 4.3: Examples of biased sloppy workers

utilizing the setting of a.

(b) Then set $a_1 = a_0/2$, calculate the estimated truths and performance metrics using a_1.

(c) Set a_0 equal to a_1. Keep repeating (b) step until there is no significant difference between the calculated metrics.

(2) For multivalued ordinal labeling tasks, usually the possible labels would be less than 10 in real world applications, the typical value chosen for a would be 1 as illustrated in our work.

(3) Another way to decide a in some cases is to request an expert to define the value regarding the task. For example, while labeling side effect level of a new type of medicine, a doctor who is highly skilled in similar medical domain should be defining the value of a.

For the purpose of easier explanation and experimentation, we set a just as 1 in the context of the paper.

The curve shown in Figure 4.2 indicates a sloppy worker with the error (deviation from true labels) perfectly evenly distributed between -1 and 1. That means, the probability of worker gives error as -1 is exactly same as the probability error as 1. We call this type of worker as perfectly erred sloppy worker. As an example, a perfectly erred sloppy worker j might have probability of 0.6 giving label same as the true label, 0.2 for having deviation as $+1$, and 0.2 for giving deviation as -1. It is not necessary that every sloppy worker be perfectly erred. In most cases, the probability of sloppy worker showing error as -1 is not equal to probability of error as 1. For example, the possibility of deviation as -1 could be greater than probability of deviation of $+1$. Figure 4.3 presents examples for this type of sloppy workers. The curve as "sloppy worker_1" give higher probability for deviation as 1 than -1, and "sloppy worker_-1" shows the worker tends to have deviation as -1 more than as 1. We could further divide the unevenly erred sloppy workers as follows:

(1) Randomly erred sloppy worker: This type of sloppy workers might have probability of giving deviation on one side slightly more than the other. For example, a worker j has the probability

of 0.4 for providing label same as the true label, 0.4 for having deviation as +1, and 0.2 for deviation as −1. Randomly erred sloppy workers are quite similar to perfectly erred sloppy workers as to the amount of information they could provide. Thus in our experimentation section, we only give the results of the influence of perfectly erred sloppy workers on different models, such as majority voting, GLAD algorithm, optimization based truth discovery method, etc.

In our work, The randomly erred and perfectly erred sloppy workers are not removed while applying aggregating algorithms. One reason is that both of their errors are within the fault tolerance range $[-a, +a]$, thus they do not affect the estimated truths much. In addition, their weights in aggregating algorithms would be much lower compared to weights of reliable workers, and corrected biased sloppy workers. Overall, the labels of randomly and perfectly erred sloppy workers would have small influence on approximating the truths.

(2) Biased sloppy worker: Different from randomly erred sloppy workers, we are able to extract useful information from the biased sloppy workers. This type of sloppy worker is the same as the biased annotator mentioned in [129]. Our proposed algorithm focuses on identifying the biased sloppy workers. The reasons we do not deal with the perfectly erred sloppy workers are: (1) it is difficult to extract information from their provided labels. (2) It could be possible to cancel some errors between this type of worker to some extent, by just utilizing an average aggregation algorithm (proved in theoretical in Section 4.3.1.2). The reasons also apply to randomly erred sloppy workers. The confusion matrix listed in Table 4.2(b) is an example of a biased sloppy worker. We can further divide the workers into two sub-categories: Positive biased sloppy workers, who tend to give much higher probability for deviation as +1 comparing to −1. Negative biased sloppy workers. They biased more towards −1 rather than +1.

Table 4.3 gives the summarization of the definitions of different types of workers discussed before. The statistic traits column shows the characteristics we used in this work for categorizing the crowd. To separate the randomly erred sloppy workers from the biased sloppy workers, a threshold value could be defined. A similar method while fixing the value of a for fault tolerance range could be utilized here: We first set the threshold value as 0.5 to recognize the randomly erred workers from biased sloppy workers. Then binary search a value within range $[0.5, 0.9]$ as the threshold till convergence. Another way could be easily implemented to approximate the threshold. We call it as the incremental heuristic way: test the values in $\{0.5, 0.6, 0.7, 0.8, 0.9\}$ on the collected label set,

TABLE 4.3: Definitions of different types of workers

<table>
<tr><th colspan="4">Worker Types</th><th>Definition</th><th>Statistic Traits</th></tr>
<tr><td colspan="4">Reliable Worker</td><td>Worker who provide high quality labels</td><td>Accuracy>0.5</td></tr>
<tr><td rowspan="5">Noisy Worker (Provide low quality labels)</td><td colspan="3">Spam Worker</td><td>Worker who provide useless labels</td><td>Accuracy≤0.5, and having deviations outside the fault tolerance</td></tr>
<tr><td rowspan="4">Sloppy Worker (Make errors which are mostly equal to +1 or -1)</td><td colspan="2">Perfectly Erred Sloppy</td><td>The errors evenly distributed between +1 and -1</td><td>Accuracy≤0.5, and Pr(error=1) equals to Pr(error=-1)</td></tr>
<tr><td colspan="2">Randomly Erred Sloppy</td><td>The prob. of errors between observed labels and true labels on one side (e.g. +1) is slightly different from the other (e.g. -1)</td><td>Accuracy≤0.5 and 0≤Pr(error=1)≤0.5 and 0≤Pr(error=-1)≤0.5</td></tr>
<tr><td rowspan="2">Biased Sloppy</td><td>Positive Biased</td><td>The distribution of deviation between observed labels and true labels as +1>as -1</td><td>Accuracy<0.5 and Pr(error=1)>0.5</td></tr>
<tr><td>Negative Biased</td><td>The distribution of deviation between observed labels and true labels as +1<as -1</td><td>Accuracy<0.5 and Pr(error=-1)>0.5</td></tr>
</table>

separately. The one which gives the best performance is chosen as the threshold. The incremental method is computationally less complex for obtaining the threshold value compared to the former one, and is efficient for some datasets.

4.3.1.2 Formalization of sloppiness

We define the mathematical notations for the problem of making use of sloppy workers′ labels. Suppose there are N items which need to be labeled and M workers for the task. Let $L = \{1, 2, ..., C\}$ be the set of labels in which workers could choose from, $y_i^j \in L$ is the label assigned to i^{th} item by worker j, and y_i is the true label of item i. $\hat{y_i}$ is the estimated true label for item i, which is called the consensus label. Assume $\varepsilon(= y_i^j - y_i)$ is the deviation/error between the observed label (obtained from workers) and true label. Table 4.4 summarizes the important notations used in the paper. Define the following probabilities:

$$P_\varepsilon^j = Pr(y_i^j = c + \varepsilon | y_i = c) \tag{4.4}$$

According to the definition of a perfectly erred sloppy worker, we know that $\varepsilon \in \{0, 1, -1\}$, and $P_1^j = P_{-1}^j$, where,

$$P_1^j = Pr(y_i^j = c + 1 | y_i = c) \tag{4.5}$$

$$P_{-1}^j = Pr(y_i^j = c - 1 | y_i = c) \tag{4.6}$$

$$P_0^j = Pr(y_i^j = c | y_i = c) \tag{4.7}$$

TABLE 4.4

Frequently used notations

Symbol	Definition
M	number of workers
N	number of items
L	set of labels worker could choose from
$Y^{(*)}$	true labels set of all items
$Y^{(j)}$	set of labels obtained from worker j
$Y^{'(j)}$	set of labels after bias correction from worker j
W	the set contains all worker weights
Θ	the set contains all worker accuracies
$E(\hat{y}_{ic})$	expected consensus label for item i, whose true label is c
y_i	true label of item i
y_i^j	label assigned to item i by worker j
$\hat{y}_i$	consensus label for item i
ε_i^j	deviation between label from worker j and truth for item i
ω_j	weight of worker j
y_{ic}^j	label assigned by worker j to item i, whose true label is c
P_ε^j	probability of worker j has deviation from true label as ε
θ_j	accuracy of worker j
α_j	conditional probability of worker j has error as $+1$
β_j	conditional probability of worker j has error as -1
η_a	marginal error threshold on worker is highly biased or not
η_b	bias score threshold for biased workers

$$\sum_{\varepsilon\in\{-1,0,1\}} P_\varepsilon^j = 1 \quad and \quad 0 \le P_{-1}^j, P_0^j, P_1^j < 0.5 \tag{4.8}$$

We first give proof that under some conditions, the existence of perfectly erred sloppy workers have no significant influence on the consensus labels. Let the calculated expectation of consensus label for item i be $E(\hat{y_{ic}})$. Here we also give assumption that the estimated labels are obtained by averaging the observed labels from workers, and every worker is independent from each other. Also, suppose the true label for item i as c, which means $y_i = c$. Thus,

$$\begin{aligned} E(\hat{y}_{ic}) &= E[\sum_{j=1}^{M} y_{ic}^j/M] = \frac{1}{M}E[y_{ic}^1 + y_{ic}^2 + ... + y_{ic}^M] \\ &= \frac{1}{M}(E[y_{ic}^1] + E[y_{ic}^2] + ... + E[y_{ic}^M]) \end{aligned} \tag{4.9}$$

where y_{ic}^j is the label given to item i whose true label is c, and $\hat{y}_{ic}$ denotes the estimated label for item i, for which the true label is c. The $E[y_{ic}^j]$ is calculated as:

$$
\begin{aligned}
E(y_{ic}^j) &= \sum_{k\in\{c-1,c,c+1\}} [y_{ic}^j \times Pr(y_{ic}^j = k)] \\
&= (c-1) \times P_{-1}^j + c \times P_0^j + (c+1) \times P_1^j \\
&= c - (P_{-1}^j - P_1^j) \quad or \quad c + (P_1^j - P_{-1}^j)
\end{aligned} \tag{4.10}
$$

The formula (4.10) is obtained based on the condition $\sum_{\varepsilon\in\{-1,0,1\}} P_\varepsilon^j = 1$ in (4.8). The hypothesis here is that worker j is a perfectly erred sloppy worker, which indicates that $P_1^j = P_{-1}^j$. The result could be obtained as $E(y_{ic}^j = c)$. We then could get $E(\hat{y_i})$ in (4.9):

$$
E(\hat{y}_{ic}) = \frac{\sum_{j=1}^{M} c}{M} = c \tag{4.11}
$$

Till now, we proved that existence of perfectly erred sloppy workers have no significant influence on the expected consensus labels, under the conditions that workers are independent from each other and using the average aggregating algorithm. However, there are two problems in real world applications:

(1) Mostly, it is too ideal to have perfectly erred sloppy workers: a worker should perfectly avoid the true grades, and evenly distribute labels only one scale up and down around the true grades. In real world scenarios, the sloppy workers would probably have a higher possibility towards either one of the errors (ε).

(2) It is unrealistic to obtain the exact expected consensus label as indicated in formula (4.9) and (4.11): while inferring equation (4.11) from (4.9), we first calculated $E(y_{ic}^j)$, which is c for perfectly erred sloppy workers. Only when a worker has labeled the same task for a large enough time, could we claim the expectation value. However, in many applications, only one label is provided from each worker on the same item. We call this fact "worker-item-uniqueness". Due to this uniqueness, the final aggregated label usually is not as good as what we expected.

These problems lead to the difficulty of making use of perfectly erred workers, since they are randomly biased and we can't cancel the errors inside the labels. However, it is possible to use the labels of the biased sloppy workers. Passonneau and Carpenter [129] proposed and validated in their work that a highly biased worker, although inaccurate, can provide as much information as a more accurate labeler. They also mentioned that weighted voting schemes are not proper approaches to infer true labels. We first examine prevalent weighted voting approaches to show why this is the

case. From the definition, we could see that an observed label y_i^j from a sloppy worker equals to the sum of true label y_i of item i and ε_i^j. Thus, the generic form for weighted learning is presented as following:

$$\hat{y}_{ic} = \sum_{j=1}^{M} \omega_j (y_i + \varepsilon_i^j) = \sum_j \omega_j y_i + \sum_j \omega_j \varepsilon_i^j, \quad where \quad \sum_j \omega_j = 1 \tag{4.12}$$

where ε_i^j is the bias made by worker j on item i, ω_j denotes the weight of worker j, and it also represents the reliability of the worker. In state-of-art approaches, ω_j is usually set proportional to worker's accuracy, such as the inversion of worker's variance $1/v_j$. In truth discovery algorithms, the weights are chosen based on the deviation function (loss function) between true grades and observed grades. In this case, the worker's weight is actually still proportional to accuracy. These methods give more weight to workers with higher $Pr(y_i^j = c | y_i = c)$.

As we know, biased workers usually are inaccurate, which means their labels have low accuracy. In order to deal with the scenarios with highly biased sloppy workers, we propose the iterative self-correcting approach to estimate the worker reliability and true labels.

4.3.2 Bias detection on sloppy worker

Before we explain the details of the iterative self correcting algorithm, we first introduce some terms and measurement metrics. For each worker j, assume the accuracy of j as θ_j, and it is calculated as $\theta_j = Pr(y_i^j = y_i) = \sum_{i=1}^{N} Pr(y_i^j = c | y_i = c) \times Pr(y_i = c)$. If we know the truth labels for all the items, we can try to estimate the worker's accuracy with the ratio $\theta_j = a/(a+b)$, where a is the number of correct and b is the number of incorrect answers from the worker. Since the workers are assumed to make errors between $+1$ and -1, we use α_j and β_j to denote the conditional probability of j has error/deviation as $+1$ and -1 respectively. In other words, we assign parameters to each worker: if this worker makes errors while labeling an item, the probability of getting deviation $+1$ is α_j, and deviation as -1 is β_j.

$$\begin{aligned} \alpha_j &= Pr(\varepsilon = 1 | y_i^j \neq y_i) \\ \beta_j &= Pr(\varepsilon = -1 | y_i^j \neq y_i) \end{aligned} \tag{4.13}$$

It is also true that $\alpha_j + \beta_j = 1$. When $\alpha_j >> \beta_j$, it indicates the worker tends to give positive deviations. While if $\beta_j >> \alpha j$, the worker is more likely to have negative deviations. Similarly,

once we know the estimated true labels set, we could approximate α_j and β_j. Due to the sparsity problem while approximating all these parameters, it is necessary to fix another way to estimate θ_j, α_j and β_j. Based on the work of [134] and [125], we use vanilla Bayesian estimation strategy for estimating them. We treat θ_j as a distribution, and presume the conjugate prior for $Pr(\theta_j)$ as uniform distribution in the $[0, 1]$ interval. Thus, after collecting a correct and b incorrect answers from the worker, the posterior probability $Pr(\theta_j)$ follows a Beta distribution $B(a+1, b+1)$:

$$Pr(\theta_j) = [\theta_j]^a [1-\theta_j]^b \frac{1}{B(a+1, b+1)} \tag{4.14}$$

Similarly, for each worker j, we set the prior distribution over the bias (α_j and β_j) as uniform distribution. Then the posterior would be a Beta distribution with hyper–parameter $b = (b_1, b_{-1})$ for α_j and β_j, where b_1 is the prior $+1$ bias count, and b_{-1} is the prior -1 bias count.

$$Pr(\alpha_j) = [\alpha_j]^{b_1} [\beta_j]^{b_{-1}} \frac{1}{B(b_1+1, b_{-1}+1)} \tag{4.15}$$

Since $\alpha_j + \beta_j = 1$, so $\beta_j = 1 - \alpha_j$, then we get:

$$Pr(\alpha_j) = [\alpha_j]^{b_1} [1-\alpha_j]^{b_{-1}} \frac{1}{B(b_1+1, b_{-1}+1)} \tag{4.16}$$

To measure the highly biased worker's reliability, we define a bias score for each of them. Higher bias score indicates the worker with higher bias, thus the more information he/she would be able to provide as those accurate workers. The bias score (BS) is derived from the information gain:

$$BS(\alpha_j) = H(bias|0.5) - H(bias|\alpha_j) \tag{4.17}$$

where $H(bias|0.5)$ denotes the entropy of choosing error as $+1$ with probability of 0.5, and -1 with probability of 0.5. $H(bias|\alpha_j)$ represents the entropy for the specific worker j with error probability α_j and β_j. Since $\beta_j = 1 - \alpha_j$, we present the entropy based on α_j.

$$\begin{aligned} H(bias|0.5) &= -[0.5log0.5 + 0.5log0.5] = -log0.5 \\ H(bias|\alpha_j) &= -[\alpha_j log\alpha_j + (1-\alpha_j)log(1-\alpha_j)] \end{aligned} \tag{4.18}$$

When $\alpha_j = 0.5$, then $\beta_j = 0.5$, $BS = 0$. It indicates that the worker is not biased. In this case, the worker is actually perfectly erred. When $\alpha_j \to 1$ or $\beta_j \to 1$, $H(bias|\alpha_j) = 0$, thus $BS = log2$.

Due to the fact that we treat the worker's error/bias as distribution, the expected $BS(\alpha_j)$ can be calculated as (α_j is a random variable):

$$E(BS(\alpha_j)) = \int Pr(\alpha_j) \cdot BS(\alpha_j) d\alpha_j \tag{4.19}$$

We know $Pr(\alpha_j)$ from equation (4.16). After some algebraic manipulations for equation (4.19), we could obtain the expectation of $BS(\alpha_j)$ as:

$$E(BS(\alpha_j)) = log2 - \Psi(b_1 + b_{-1} + 1) + \frac{b_1 \Psi(b_1 + 1) + b_{-1} \Psi(b_{-1} + 1)}{b_1 + b_{-1}} \tag{4.20}$$

where $\Psi(x)$ is the digamma function. By setting a threshold η_b, we could correct the highly biased workers with $E(BS(\alpha_j)) > \eta_b$. However, it is also necessary to take into account the worker's accuracy, before we correct the bias/error. This is because the error α_j and β_j are assumed as conditional probability in this work. The true error probability (marginal probability) of a worker will depend also on the accuracy of the worker. For example, if a worker's accuracy is 0.9, thus the possibility of the worker making errors is only at 0.1. Although it is possible that the worker might have a 0.9 possibility with +1 error for any item, we shouldn't correct his labels (by subtracting 1). If we do so, all the items with correct answers will be changed also, thus degrade the final results.

In order to determine whether to correct bias behavior of a worker, we proceed in the following way: As mentioned before, the worker's accuracy θ_j is assumed as a distribution, and set Beta distribution as its conjugate prior. Thus, the error rate of worker is $1 - \theta_j$. We give the posterior mean of the marginal probability of errors, which is the best point estimation for $\alpha_j(1-\theta_j)$ and $\beta_j(1-\theta_j)$: $E(\alpha_j(1-\theta_j)) = \int_{\theta_j=0}^{1} \int_{\alpha_j=0}^{1} ((1-\theta_j) \times Pr(\theta_j))(\alpha_j Pr(\alpha_j)) d\theta_j d\alpha_j$, so:

$$E(\alpha_j(1-\theta_j)) = [1 - \frac{a+1}{a+b+2}] \cdot \frac{b_1 + 1}{b_1 + b_{-1} + 2} \tag{4.21}$$

Similarly, estimation for $E(\beta_j(1-\theta_j))$ could be obtained.

By setting a threshold for the expected error rate of the worker as η_a, we would be able to determine whether to take into account the bias behavior. Only if $E(\alpha_j(1-\theta_j)) > \eta_a$ or $E(\beta_j(1-\theta_j)) > \eta_a$, will we then consider correcting errors made by the workers based on their bias score. The η_a could be chosen through the incremental heuristic way or binary search approach, both were presented while choosing the threshold for separating the randomly erred sloppy workers from biased workers. The value which has the best performance will be selected as the threshold η_a.

Algorithm 4.1 Iterative self-correcting truth discovery algorithm.

Input: Observations from M workers: $\{Y^{(1)}, Y^{(2)}, ..., Y^{(M)}\}$, threshold η_a, and η_b.
Output: Estimated true labels for N items $Y^{(*)} = \{y_i\}_{i=1}^{N}$, worker weights $W = \{\omega_j\}_{j=1}^{M}$, estimated worker accuracies set $\Theta = \{\theta_j\}_{j=1}^{M}$, and workers bias correcting factor B.

Initialize the truths $Y^{(*)}$;
repeat
    Remove spam workers first from the crowd;
    for $j = 1...M$ **do**
        $a \leftarrow$ number of correct labels
        $b \leftarrow$ number of incorrect labels
        $b_1 \leftarrow$ number of $+1$ error ($y_i^j - yi$ as $+1$)
        $b_{-1} \leftarrow$ number of -1 error ($y_i^j - y_i$ as -1)
        Calculate $E(BS(\alpha_j))$ from equation (4.20)
        if $E(BS(\alpha_j)) < \eta_b$ **then**
            $Y^{'(j)} \leftarrow Y^{(j)}$
            Continue
        else
            Calculate $max(E(\alpha_j(1-\theta_j), E(\beta(1-\theta_j)))$
            if $maximum > \eta_a$ **then**
                if $\alpha_j > \beta_j$ **then**
                    {Correcting $Y^{'(j)}$ by subtracting 1}
                    $Y^{'(j)} \leftarrow Y^{(j)} - 1$
                else
                    {Correcting $Y^{(j)}$ by adding 1}
                    $Y^{'(j)} \leftarrow Y^{j} + 1$
                end if
            end if
        end if
    end for
    {Calculate the worker's weight}
    Update worker's weight W using equation (4.24) to infer worker reliability after bias correction, based on the estimated truths.
    {Calculate the estimated truths}
    for $i = 1...N$ **do**
        Update the truth of i^{th} item y_i based on observations and current weight estimations, from the worker who contributed to this item, according to equation (4.25).
    end for
until Convergence
{Calculate the worker's bias}
for $j = 1...M$ **do**
    Compare the $Y^{'(j)}$ and $Y^{(j)}$, get the bias of worker j, insert the bias into B.
end for
Return $Y^{(*)}$, W, Θ, and B

4.3.3 Iterative self correcting framework

The iterative self-correcting algorithm is detailed in Algorithm 4.1. The algorithm is based on the principle of optimization based truth discovery models: If a worker provides trustworthy labels frequently, he will be assigned a high reliability; if a label is provided by a reliable worker, it will have higher probability to be chosen as truth. As Li et al. [132] concluded in their work, the principle of truth discovery is captured through the optimization formulation in Eq. (4.2).

The proposed framework is an optimization based method for bias corrected observations.

In Algorithm 4.1, line 4-25 represents the process of detecting and correcting the biased sloppy workers. After obtaining the de-biased answers $Y^{'(j)}$ for each worker, we discover the truths through the following formulation:

$$\begin{aligned} &\min_{\{\omega_j\},\{y_i\}} f(Y^{(*)}, W) = \sum_{i=1}^{N}\sum_{j=1}^{M} \omega_j d(y_i^j, y_i), \\ &where \quad y_i^j \in Y^{'(j)} \quad s.t. \sum_{j=1}^{M} exp(-\omega_j) = 1 \end{aligned} \tag{4.22}$$

where the $d(\cdot)$ is called the loss function as mentioned in [132]. Since originally the sloppy worker only err $+1$ or -1 around the ground truths, it would be same to either use 0-1 loss or (normalized) squared loss. However, after correcting the bias for the worker, it is possible that worker might err $+2$ or -2 from the truth answers. For example, if a highly biased worker has 80% of his answers with $+1$ error. According to Algorithm 4.1, all the observed answers from the worker are corrected by subtracting 1 from $Y^{(j)}$. Thus, if the worker has any answer with -1 error, then the de-biasing would result err -2 from the truths. Thus we propose to use the 0-1 loss in this case:

$$d(y_i^j, y_i) = \begin{cases} 1, & if \quad y_i^j \neq y_i \\ 0, & otherwise \end{cases} \tag{4.23}$$

To learn the truth answer set Y and worker's weights and biases, we optimize the objective function in Eq. 4.22 by applying the block coordinate descent approach [135]. The approach iteratively conducts two-step procedure until convergence:

(i) Update worker weights while fixing truths of the items $Y^{(*)} = \{y_1, y_2, ..., y_N\}$: In this step, the true label for each item is fixed, we estimate the worker weights W based on the difference between true label and corresponding observed answer. The weight of each worker is calculated through formula (4.28).

$$W \leftarrow \arg\min_{W} f(Y^{(*)}, W) \quad s.t. \sum_{j=1}^{M} exp(-\omega_j) = 1 \tag{4.24}$$

(ii) Update the truths y_i while fixing the worker weights and every other item's truths $\{W, y_1, ..., y_{i-1}, y_{i+1}, ..., y_N\}$: In this step, the true label of each item is updated by minimizing the weighted differences between the true labels and observations.

$$y_i \leftarrow \arg\min_{y_i} f(Y^{(*)}, W) \tag{4.25}$$

This two-step approach corresponds to line 26-31 in Algorithm 4.1. The step (i) is the worker weight updating process from line 26 to 27. Line 28-31 is the estimated truths updating procedure as described in step (ii).

To derive the source weight in step (i) of the block coordinate descent method, the Lagrange multiplier approach can be utilized on optimization problem in (4.22):

$$L(W, \lambda) = \sum_{j=1}^{M} \omega_j \sum_{i=1}^{N} d(y_i^j, y_i) + \lambda(exp(-\omega_j) - 1) \tag{4.26}$$

By taking the partial derivative with respect to ω_j be 0, it would be able to obtain:

$$\lambda = \frac{\sum_{i=1}^{N} d(y_i^j, y_i)}{exp(-\omega_j)} \tag{4.27}$$

From the constraint, we know that $\sum_{j=1}^{M} exp(-\omega_j) = 1$, thus,

$$\omega_j = -log \frac{\sum_{i=1}^{M} d(y_i^j, y_i)}{\sum_{j'=1}^{M} \sum_{i=1}^{N} d(y_i^{j'}, y_i)} \tag{4.28}$$

In order to give an overview of the proposed framework, we present the diagram of the procedure in Figure 4.4. We first initialize the truths for all the items. The initialization values are randomly selected from the prior uniform distribution we assumed for the truths. Then we iteratively estimate the true labels as indicated in Algorithm 4.1, line 2-32. The spam detection approach used in our work for real world dataset is based on the mean squared error measurement. The algorithm is adopted from the work of Vuurens et al. [60]. The authors proposed a function called *RandomSep* to recognize the workers who provide random labels. The *RandomSep* is defined as:

$$RandomSep = \frac{\sum_{v \in V} \varepsilon_r^2}{|V|} \tag{4.29}$$

where ε_r is the error which represents the ordinal difference between the label a worker given and the estimated true label. V is the collection of all labels offered by each worker. By setting threshold for *RandomSep*, we could detect the spam worker. For example, through setting the threshold value as 1, we could filter out the workers with $RandomSep > 1$, and leave the workers only err $+1$ or -1 around the estimated truths.

The bias correcting process corresponds to the line 4-25 in Algorithm 4.1. Finally, the labels after spam removing and bias correction are used to estimate the truths through coordinate descent algorithm (line 26-31 in Algorithm 4.1).

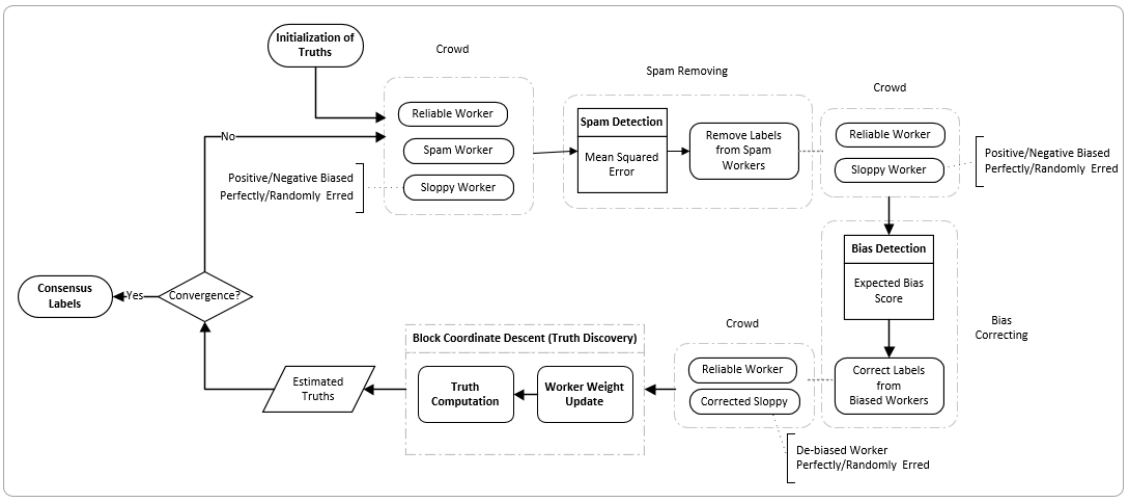


Figure 4.4: Diagram of the proposed ITSC-TD framework

4.4 Experimental results and analysis

This section presents experimental results and analysis of the proposed ITSC-TD framework on synthetic and real world datasets. Section 4.4.2 presents the results on synthetic datasets to show the influences of different types of sloppy workers on a controlling environment. Section 4.4.3 gives experiments on two publicly available real world datasets to show the effectiveness of the proposed methodology in obtaining truths from noisy crowdsourced data. We also implement a set of state-of-art models, which are used as comparison methods, in order to demonstrate the efficacy of the proposed ITSC-TD approach. These comparison models are presented in Section 4.4.1.2. The evaluation metrics and the implemented comparison methods are discussed in Section 4.4.1.

4.4.1 Experimental setup

4.4.1.1 Performance measures

In the experiments, the inputs for the models are observations for the items from crowd workers. The aggregating algorithms are applied to output the estimated truths and worker weights. Our proposed methodology and some of the state-of-art methods are conducted in an unsupervised manner, and the ground truths will be only used for evaluating the performance. Here, we focus on categorical type of data, thus we adopt the following measures.

- Accuracy: accuracy is computed as the percentage of the approach's output that are same as the ground truths.

- F1 measure: it is a weighted average of precision and recall of an approach. F measure is different from the accuracy, which equally weights the positive and negative results.

Accuracy and F1 measure how well the model estimates the truths. The higher the measures, the closer the method's outputs to the ground truths, and thus the better it performs.

4.4.1.2 Proposed and comparison methods

The proposed ITerative Self-Correcting Truth Discovery method in the experiment is denoted as ITSC-TD. We compare our model with the following methods which are designed to infer truths and worker weights in crowdsourcing systems:

Majority Voting (MV) which simply takes the majority label as the truth for an item. If there is a tie while applying MV, we randomly select an answer from the voted results in our work. Expectation Maximization [46] (EM) model proposed by Dawid and Skene (EM-DS) models a confusion matrix for each worker, and uses EM to estimate the true labels. GLAD [45] models the expertise of each worker through a single parameter, and estimates truths via the EM algorithm. Optimization based Truth Discovery (TD) [132] is also a weighted voting scheme to infer the true labels.

We implement all the comparison methods, and all of the parameters in the models are set according to corresponding papers. The experimental results are compared using different data sets. In the proposed ITSC-TD methodology, we need to set the accuracy (η_a) and Bias Score (BS) threshold (η_b) to determine a highly biased sloppy worker. Here, we initially set the η_a equals to 0.5, and α_j as 0.5 to obtain η_b. Then 10 values are selected randomly for η_a and α_j respectively in the range [0.5, 0.7], for parameter sensitivity analysis. The final results are obtained through averaging the 10 results of the settings.

4.4.2 Experiments on synthetic data set

In the this section, we focus on experimenting with the influence of sloppy workers to the quality of estimated truths. The synthetic data is simulated based on the work of Alfaro and Shavlovsky [7]. We consider 50 workers and 50 items in the simulated environment. Assume each item is labeled by 6 workers, and the labels are chosen from the scale: $\{1, 2, 3, 4, 5\}$. Suppose we would like to simulate the grading process for a set of students$'$ submissions, which are the items, in class. A latent variable model is utilized to approximate the gold truths: let the true quality of each item i be denoted as y_i. To simulate all the y_i, we assume the existence of a real-valued latent

variable q, where q is normally distributed with mean $\mu = 0$ and standard deviation $\sigma = 1$. The variable y_i results from an "incomplete measurement" of q, where one only determines the interval into which q falls:

$$y_i = \begin{cases} 1, & if \quad q < \mu - 2\sigma \\ 2 \quad or \quad 3 \quad or \quad 4, & if \quad q \in [\mu - 2\sigma, \mu + 2\sigma] \\ 5, & if \quad q > \mu + 2\sigma \end{cases} \tag{4.30}$$

In equation (4.30), while $q \in [\mu - 2\sigma, \mu + 2\sigma]$, y_i could choose from 2, 3, and 4. In order to assign one of the labels to y_i, we assume another latent real-valued variable u which is uniformly distributed on $[0, 3]$. The label of y_i is determined as follows:

$$y_i = \begin{cases} 2, & if \quad u \in [0, 1) \\ 3, & if \quad u \in [1, 2) \\ 4, & if \quad u \in [2, 3] \end{cases} \tag{4.31}$$

Combining equation (4.30) and (4.31), we could obtain the formula for determining y_i:

$$y_i = \begin{cases} if \quad q < \mu - 2\sigma, \quad 1 \\ if \quad q \in [\mu - 2\sigma, \mu + 2\sigma], \quad \begin{cases} if \quad u \in [0, 1), \quad 2 \\ if \quad u \in [1, 2), \quad 3 \\ if \quad u \in [2, 3], \quad 4 \end{cases} \\ if \quad q > \mu + 2\sigma, \quad 5 \end{cases} \tag{4.32}$$

Two types of workers are simulated: reliable and sloppy worker. Each worker j was assigned a specific accuracy. For reliable workers, the accuracies are uniformly distributed on $(0.5, 0.9]$. For sloppy workers, their accuracies are uniformly distributed on $[0.1, 0.5]$. It is also necessary to allocate a bias rate for sloppy workers. The bias rate is the percentage of positive (negative) errors existing among incorrect answers for positive (negative) biased worker. A perfectly erred sloppy worker should have a bias rate as 0.5. The items are then randomly distributed to workers for labeling: each item is required to be labeled by 6 workers, and each worker needs to label 6 items. The observations are labels obtained from workers are simulated in the following way:

- If the worker i is reliable: according to the accuracy of i, determine whether to assign true label

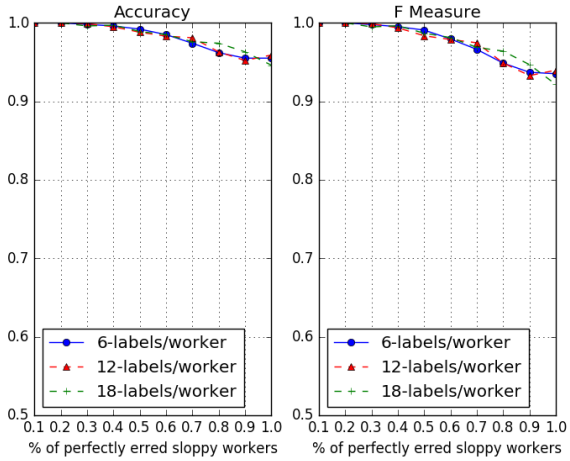


Figure 4.5: Influence of perfectly erred sloppy workers with multiple labels/worker on consensus labels

to the item or not. If an incorrect answer should be given, randomly select a label one scale up or down (if possible) of the true label for the item.

- If the worker i is sloppy: accuracy is still first utilized to decide to provide a correct or incorrect answer. If a wrong answer is supposed to be given, the sloppy worker type (positive, negative or perfectly erred) and bias rate are then applied to provide the observed label.

Once all the simulated observations are generated, we are able to apply the aggregating models to estimate truths and worker weights from them. For each setting, the data is simulated through 100 runs, and the results are reported as the average over the 100 runs.

4.4.2.1 Simulation on perfectly erred sloppy workers

First of all, we investigate the perfectly erred sloppy workers′ influence on the consensus results in ideal occasions. Assume each worker provides multiple labels (k) for the item which is assigned for them. Three different settings for k were experimented here: 6, 12, and 18 labels were generated by each worker. That means, if item i is assigned to worker j to get observed labels, j is responsible to provide k labels for i. We assume that workers are independent from each other, and the k labels from the same worker are also independently given. Figure 4.5 shows the outcome. The consensus labels are given through averaging all the observations. In Figure 4.5, the x axis represents the proportion of the perfectly erred sloppy workers among the crowd, and y axis shows accuracy and f1 measure respectively. Although the metrics start decreasing at the point of 0.5 for

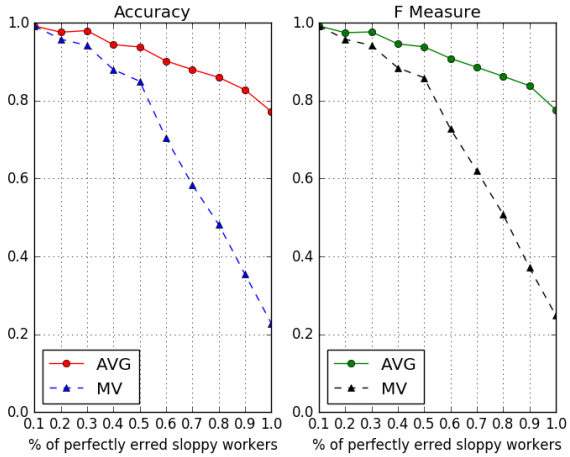


Figure 4.6: Influence of perfectly erred sloppy workers with one label/worker on consensus labels

the proportion of sloppy workers, we could still obtain high quality results. For example, at the worst case scenario, with proportion equals to 1, the accuracy is > 0.95, and f1 measure > 0.9. The results proved that given perfectly erred sloppy workers, under the specific conditions, it is still possible to obtain high quality consensus results, as we mentioned in Section 4.3.1.2.

Due to the "worker-item-uniqueness", it is necessary to investigate the influence of sloppy workers on aggregating results with only one observation per worker for item i. Figure 4.6 presents the metrics calculated for the results with single label provided per worker for one item. Comparing to the accuracies and F1 measures with average (AVG) as the aggregating algorithm in Figure 4.6 and 4.5, we can see that without multiple labels per worker, the perfectly erred sloppy workers degrade the aggregated results greatly. The reason that performance in Figure 4.6 is worse than Figure 4.5 is: while a worker independently provides multiple labels/item, the sloppiness within their labels is most likely canceled with each other (Probability of negative error=Probability of positive error=0.5). When only one label is given for each item/worker, the label is randomly given and independent from each worker, thus the overall probability of obtaining error as $+1$ is not necessarily equal to the probability of error -1. The averaging algorithm could cancel some of the errors among the grades, however, it is usually used for numerical data instead of categorical ordinal data. Majority voting (MV) is the simplest, and one of the prevalent aggregating methods for getting consensus labels for categorical ordinal data. The dotted line with triangle sign in Figure 4.6 gives results of majority voting. MV showed even worse results compared to AVG method. While setting the percentage of perfectly erred workers as 100%, the difference of accuracy between AVG and MV could reach around 60%.

TABLE 4.5: Accuracy and F measure calculated for different proportions of positive biased sloppy workers

Method	Metric	Proportion of Positive Biased Sloppy Workers									
		10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
MV	Accuracy	0.98	0.96	0.92	0.88	0.82	0.62	0.46	0.34	0.24	0.20
	F1	0.98	0.97	0.93	0.89	0.84	0.63	0.52	0.38	0.25	0.23
TD	Accuracy	1.00	0.96	0.96	0.94	0.86	0.66	0.52	0.36	0.26	0.24
	F1	1.00	0.97	0.97	0.95	0.87	0.68	0.57	0.40	0.27	0.27
GLAD	Accuracy	0.98	0.96	0.94	0.94	0.92	0.66	0.56	0.38	0.26	0.22
	F1	0.98	0.97	0.95	0.93	0.92	0.68	0.60	0.38	0.27	0.24
EM-DS	Accuracy	0.98	0.96	0.98	0.95	0.93	0.70	0.60	0.38	0.26	0.24
	F1	0.98	0.96	0.98	0.95	0.94	0.72	0.60	0.42	0.26	0.24
ITSC-TD	Accuracy	**1.00**	**0.98**	**0.98**	**0.96**	**0.95**	**0.76**	**0.74**	**0.50**	**0.36**	**0.34**
	F1	**1.00**	**0.99**	**0.99**	**0.96**	**0.96**	**0.80**	**0.76**	**0.56**	**0.39**	**0.35**

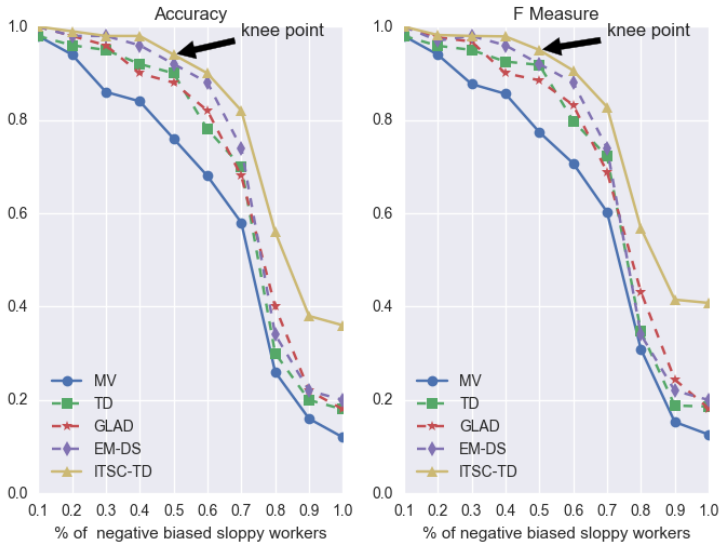


Figure 4.7: Results of accuracy and F measure for different proportions of negative biased sloppy workers

4.4.2.2 Simulation on biased sloppy workers

In order to show the influence of biased sloppy workers on the consensus results, we first conduct the experiments on either one of the two types of biases. Table 4.5 gives the metrics calculated for the proposed and comparison methods for different proportions of positive biased sloppy workers. In Table 4.5, while the proportion positive biased of sloppy workers ≤ 0.5, all the listed methods give good performance regarding the selected metrics. Among the four comparison models, EM-DS method gives relatively better results compared to the other three. If we compare the accuracy and F measure of EM-DS with the proposed algorithm while the ratio of positive biased sloppy workers is within the range [0.1, 0.5], the improvement by utilizing ITSC-TD is only from around 0 to 3%. After increasing the proportion up to > 0.5, the proposed ITSC-TD method shows

TABLE 4.6: Results for various ratios of mixture of positive and negative biased sloppy workers

Method		MV		TD		GLAD		EM-DS		ITSC-TD	
Prop. of Sloppy Workers	Ratio of Positive & Negative Biased	Accuracy	F1	Accuracy	F1	Accuracy	F1	Accuracy	F1	Accuracy	F1
20%	0.5:0.5	0.96	0.96	0.98	0.97	0.98	0.98	0.98	0.97	**0.98**	**0.98**
	0.8:0.2	0.96	0.97	0.98	0.98	0.92	0.91	0.98	0.98	**1.00**	**1.00**
	0.4:0.6	0.97	0.96	0.98	0.98	0.98	0.97	0.98	0.98	**0.98**	**0.98**
40%	0.5:0.5	0.84	0.85	0.92	0.92	0.90	0.89	0.96	0.96	**0.98**	**0.98**
	0.8:0.2	0.80	0.81	0.94	0.93	0.89	0.86	0.94	0.94	**0.96**	**0.96**
	0.4:0.6	0.80	0.81	0.88	0.89	0.84	0.83	0.90	0.90	**0.95**	**0.94**
50%	0.5:0.5	0.75	0.76	0.84	0.84	0.83	0.82	0.90	0.90	**0.92**	**0.91**
	0.8:0.2	0.76	0.78	0.88	0.87	0.82	0.81	0.88	0.88	**0.94**	**0.94**
	0.4:0.6	0.72	0.77	0.80	0.83	0.78	0.80	0.86	0.86	**0.90**	**0.91**
60%	0.5:0.5	0.60	0.68	0.80	0.81	0.78	0.79	0.86	0.86	**0.92**	**0.92**
	0.8:0.2	0.58	0.57	0.78	0.78	0.69	0.67	0.74	0.74	**0.88**	**0.87**
	0.4:0.6	0.60	0.59	0.68	0.67	0.68	0.66	0.78	0.78	**0.82**	**0.80**
80%	0.5:0.5	0.46	0.49	0.56	0.58	0.54	0.55	0.58	0.58	**0.72**	**0.71**
	0.8:0.2	0.32	0.36	0.46	0.48	0.38	0.40	0.44	0.43	**0.64**	**0.64**
	0.4:0.6	0.44	0.46	0.56	0.56	0.52	0.51	0.56	0.55	**0.66**	**0.67**

significant improvement compared to other comparison models for both accuracy and F measure. As an example, 14% accuracy improvement can be obtained by applying ITSC-TD compared to EM-DS approach.

Similar process and results could be obtained while experimenting on negative biased sloppy workers. Figure 4.7 showed the calculated measurements. The solid line with the triangle symbol represents the metrics results for proposed ITSC-TD methodology. The "knee point" in Figure 4.7 is the point at the proportion of negative biased sloppy workers = 0.5, and it means while the proportion > 0.5, there is significant improvement on the calculated measurements for our proposed algorithm compared to other approaches.

The experiments above assumed the existence of only one type of biased sloppy worker: either positive biased or negative biased. In order to examine the influence of mixture of positive and negative biased sloppy workers, we do simulations on various ratios of these two types of workers. The results are presented in Table 4.6. The ITSC-TD method has the best performance among all the algorithms applied. As the results showed in Table 4.6, when the proportion of sloppy workers $> 50\%$, the proposed approach has significant difference on the performance compared to other methods. The results are consistent with what we obtained with the existence of only one type of biased sloppy worker.

By comparing the results in Table 4.5 and Table 4.6, we could see that there are differences between them even with the same proportions of sloppy workers. The explanation of the distinctions is that for each table, we utilize the approach mentioned at the beginning of Section 4.4.2 to simulate the sloppy workers through 100 runs. The average over the 100 runs is then used as final outcome, which is presented in the table. Although the percentage of the biased sloppy workers might be set

same in both of the tables, each worker's individual bias rate could be different. Thus the final results vary. For example, assume a worker has 60% of her labels positively biased, and another worker has 80% of her labels positively biased. The result by taking into account the former worker will be different from the result of using the latter one. In order to see the differences of the performance between only using positive biased workers and having both positive and negative biased workers, we could set one essential comparison method for each table. As an example, we set the GLAD approach as comparison model for Table 4.5 and 4.6. The accuracies and F measures obtained for ITSC-TD, with high proportion of biased sloppy workers, have significant improvement in contrast to the metrics calculated for GLAD in both of the tables. We could conclude that the proposed methodology is efficient in both scenarios: when there is only one type of biased workers, or both types of biased sloppy workers exist among the crowd.

4.4.2.3 Simulation on all types of sloppy workers

Before showing the results for datasets with crowd including all types of sloppy workers, we would like to first investigate the performance of the proposed approach on the worker set with the mixture of perfectly erred sloppy workers and reliable workers. Like all the other comparison models we selected, the proposed ITSC-TD method utilizes the labels given by perfectly erred sloppy workers without correcting or removing them. Figure 4.8 presents the outcome of the experiments. The proportion of the perfectly erred sloppy workers ranges from 0 to 100% as shown in x axis. From the figure, we could see that compared to other comparison approaches, the proposed method has no significant difference between them. This is due to the fact that the ITSC-TD algorithm does not tackle with perfectly erred sloppy workers.

While varying the ratios of the perfectly erred sloppy workers, we could observe the influence of this type of workers on the quality of the estimated truths, which is presented in Figure 4.8. When the percentage $< 50\%$, most of the comparison models (except MV) and our proposed methodology give estimated labels precise enough. In other words, the perfectly sloppy workers have very small influence on weighted voting aggregating algorithms with their proportion less than 50%. One reason is that they make errors within the fault tolerance range, which denotes that they give labels close enough to truths. The other reason is in weighted voting, the weights given to workers are proportional to their accuracies. The perfectly erred sloppy workers thus would have low weights assigned to them. If we change the percentage till most of the workers are perfectly erred sloppy workers ($> 50\%$), the results are greatly degraded as shown in Figure 4.8. Especially when the

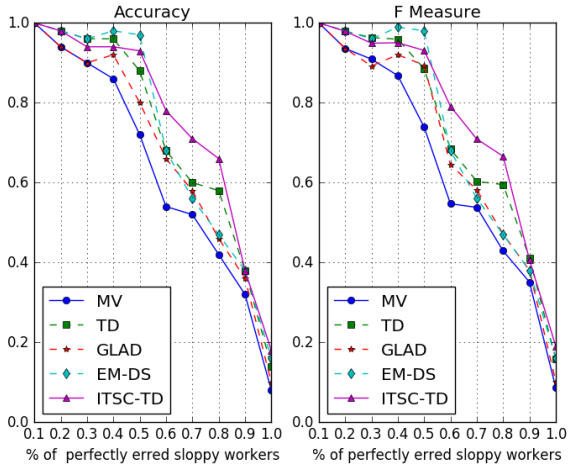


Figure 4.8: Results of accuracy and F measure for mixture of perfectly erred sloppy workers and reliable workers for proposed and comparison methods

proportion of perfectly erred sloppy workers $\geq 90\%$, all the used weighted voting algorithms' performance are almost the same as MV. In order to achieve higher accuracies, it might be useful to filter out this type of workers and remove their labels. This leads to removal of most workers from the crowd. It leaves many items remain unlabeled. Due to the fact that perfectly erred sloppy workers only have errors within the fault tolerance range, in many occasions, it is acceptable to utilize their labels to approximate the truths when there are not enough precise labels available. Based on the analysis, we do not remove the perfectly erred sloppy workers. For randomly erred sloppy workers, same conclusion could be drawn.

Next, we present the results obtained for simulations with different types of sloppy workers incorporated. Table 4.7 gives the metrics calculated for the mixture of different proportions of positive biased sloppy workers, perfectly erred sloppy workers, and reliable workers. In order to show how biased and perfectly erred sloppy workers influence the performance of the ITSC-TD algorithm, we also present results when the crowd consists of positive biased and reliable workers, as well as the crowd with perfectly erred and reliable workers. Since the results for the mixture of negative biased workers with reliable workers are quite similar to positive biased workers, we only present the calculated metrics for PB: RE.

In Table 4.7, the highlighted cells represent the methods with best performance. By comparing the experimental results, the ITSC-TD methodology shows superiority over other comparison approaches with existence of biased sloppy workers among the crowd. As an example, when the crowd contains only perfectly erred sloppy workers and reliable workers, TD algorithm gives best

TABLE 4.7: Results for the crowd with mixture of positive biased sloppy workers, perfectly erred sloppy workers, and reliable workers

Method	Metric	Ratios of different types of workers					
		PB : RE	PB : PE : RE	PE : RE	PB : RE	PB : PE : RE	PE : RE
		0.4 : 0.6	0.2 : 0.2 : 0.6	0.4 : 0.6	0.8 : 0.2	0.6 : 0.2 : 0.2	0.8 : 0.2
MV	Accuracy	0.80	0.88	0.82	0.42	0.48	0.44
	F1	0.83	0.89	0.84	0.43	0.49	0.49
TD	Accuracy	0.92	0.92	**0.96**	0.46	0.54	0.54
	F1	0.92	0.93	**0.97**	0.47	0.56	0.58
GLAD	Accuracy	0.88	0.90	0.92	0.46	0.52	0.48
	F1	0.89	0.89	0.91	0.46	0.53	0.49
EM-DS	Accuracy	0.94	**0.96**	0.94	0.50	0.60	0.52
	F1	0.93	**0.96**	0.94	0.50	0.60	0.52
ITSC-TD	Accuracy	**0.96**	**0.96**	0.94	**0.66**	**0.71**	**0.55**
	F1	**0.95**	**0.96**	0.95	**0.65**	**0.73**	**0.57**

PB: Positive Biased; RE: Reliable; PE: Perfectly Erred.

performance with PE: RE = 0.4: 0.6. Although while PE: RE = 0.8: 0.2, ITSC-TD has the largest accuracy value, only 1.9% improvement could be obtained compared to the TD approach. It could be further verified by comparing the metrics calculated for PB: RE and PB: PE: RE. For example, in Table 4.7, when the sloppy worker: reliable worker as 0.4: 0.6, if we compare the accuracy obtained for EM-DS and ITSC-TD: 2.1% improvement was achieved by applying ITSC-TD, with PB: RE = 0.4: 0.6. However, there was 0% improvement while PB: PE: RE = 0.2: 0.2: 0.6. This indicates that with the same proportion of sloppy workers, better results could be obtained with higher ratio of biased sloppy workers. The same conclusion could be drawn when sloppy worker: reliable worker as 0.8: 0.2. The reason that there is 0% improvement comparing accuracy of EM-DS and ITSC-TD, while PB: PE: RE = 0.2: 0.2: 0.6, is because no significant improvement could be obtained while the proportion of sloppy workers$<$ 0.5.

4.4.3 Experiments on real world data set

We use real world data to investigate the effectiveness of the proposed ITSC-TD methodology. Two publicly available datasets are utilized to evaluate the models, namely TREC and AdultContent2 (AC2). The original TREC dataset, which used in [112], has AMT ordinal graded relevance judgments for pairs of search queries and URLs (web pages). It consists of 98,453 ratings corresponding to 766 workers, 100 queries, and 20,232 (query, URL) pairs. The ratings were on a scale of $\{-2, -1, 0, 1, 2\}$, where -1 means missing ground truth label, and -2 corresponds to broken link. We processed this data set by filtering the ratings with value -2 and only take into account the data with gold ground truth. The final dataset contains 3275 (query, URL) instances, 722 workers and 19,699 collected labels.The ratings were mapped from $\{0, 1, 2\}$ to $\{1, 2, 3\}$. This mapping does not affect the values of accuracy and F measure, it is just for easier implementing of all the models.

TABLE 4.8: Some statistics about the TREC and AC2 datasets

	Label Levels	Instances	Workers	Ratings	Reliable	Spam	Biased	Perfectly/Randomly Erred
TREC	3	3275	722	19699	433	0	143	146
AC2	4	333	269	3324	190	19	44	16

TABLE 4.9

Comparison between different methods on TREC and AC2 dataset (without spam workers included)

	TREC		AC2	
Method	Accuracy	F1	Accuracy	F1
MV	0.476	0.481	0.763	0.741
TD	0.483	0.496	0.765	0.743
GLAD	0.498	0.501	0.761	0.739
EM-DS	0.502	0.506	**0.772**	**0.748**
ITSC-TD	**0.513**	**0.524**	0.770	0.746
Detected-Biased	96		29	

The AC2 dataset was originally used in [31], and it includes AMT judgments for websites for the presence of adult content on the page. The judgments are ordinal ratings on G, P, R, X: G for no adult content, P refers to content requires parental guidance, R means content mainly for adults, and X for hardcore porn. The original AC2 dataset consists of 97271 ratings from workers. We filter the data by only taking into account the items with gold truths. This leads to a subset of the data which consists of 3324 ratings from 269 workers for 333 websites. The ratings are mapped from $\{G, P, R, X\}$ to $\{1, 2, 3, 4\}$. The mapping has no influence on the measurement metrics. Due to the fact that we would want to deal with sloppy workers in our work, we further filter the data by choosing the ratings within the set gold truth, gold truth+1, gold truth−1. The final AC2 dataset used has 3057 ratings corresponding to 333 items, 265 workers.

There are only three levels for the rating task in TREC dataset, so we do not define and filter spam workers in this dataset. Table 4.8 gives some of the statistics for these two datasets. The "Instances" column indicates the number of items to be labeled. The last four columns present the number of different types of workers in the crowd. We give two types of experimental analysis on the real world datasets: (i) focus on dealing with the sloppy workers. In this analysis, we remove the spam workers by comparing the observed labels with gold truths. In other words, we only keep the worker judgments that are at most one level away from the gold labels. The results are presented in Table 4.9. (ii) We investigate the impact of both spam and sloppy workers on our proposed methodology. The AC2 dataset is used here for the analysis. As mentioned above, TREC dataset incorporates only three levels of labels, in which no spam worker is defined. The results are shown

TABLE 4.10

Results for AC2 dataset with both spam and sloppy workers incorporated

Method	Accuracy	F1
MV	0.756	0.725
TD	0.759	0.735
GLAD	0.752	0.731
EM-DS	0.759	0.736
ITSC-TD	**0.762**	**0.741**
Detected Spam	16	
Detected Biased	21	

as Table 4.10.

Table 4.9 summarizes the performance of all the methods in terms of accuracy and F measure on TREC and AC2 dataset without spam workers included. The last row of the table gives the number of biased workers recognized while applying ITSC-TD framework. The proposed method in our work showed the best performance compared to other comparison approaches on TREC data set. Although for AC2, EM-DS out performs the proposed methodology, there is no significant difference between all the methods applied. The different effectiveness of the proposed method on the two datasets can be explained through the statistics presented in Table 4.8. There are more biased workers (around 23%) in TREC compared to AC2 dataset (around 16%). Thus the ITSC-TD approach gives best quality outcome compared to other comparison models for TREC, while it does not show superiority in AC2 due to the low proportion of biased workers. In addition, most workers (more than 70%) in AC2 belong to reliable worker group, which makes even the MV approach already good enough for approximating the truths. Finally, the experimental results are consistent with conclusion drawn from the synthetic dataset: when the proportion of biased sloppy workers ≤ 0.5, no significant improvement could be obtained.

We next give an insight of the proposed ITSC-TD framework performance on the dataset where both spam and sloppy workers exist. As mentioned earlier, the AC2 dataset is utilized for the analysis. To detect the spam workers, a mean squared error based function, which is explained in Section 4.3.3, is applied. The results are presented in Table 4.10. The last two rows of the table give the number of spam workers and biased sloppy workers recognized in ITSC-TD framework. Different from the calculated metrics indicated in Table 4.9 for AC2, in which EM-DS showed superior performance, the proposed ITSC-TD algorithm offered the most precise estimated truths here. It is also clear that the overall quality of the estimated labels is lower in Table 4.10 than Table 4.9. The reason for the differences is that the spam workers are included in the experiments here

for Table 4.10. Due to the fact that ITSC-TD approach deals with both spam and biased workers, we could obtain better results by utilizing the proposed method than the comparison models.

4.5 Conclusion and Chapter summary

In this chapter, we present a way to tackle the challenge of obtaining true information from noisy crowdsourced data. In particular, we deal with the sloppiness in a crowd scoring task, which represents the phenomena of observed labels which fluctuate around the true labels, in order to achieve high quality estimated labels. The main contributions of the work in this chapter are as follows:

(1) Different from many research which focuses on binary labeling problems, we highlight obtaining high quality estimated true labels from the crowdsourced ordinal labeling tasks. Crowdsourcing research usually targeted on simple problems such as choosing either 0 or 1 in a task. It is not always the case in real world. We concentrate on more complicated tasks with ordinal labeling. This type of task is much more like a scoring problem in which answers are chosen from a scale which consists of multiple ordinal labels.

(2) A hierarchical categorization of the crowd workers is introduced. The workers who provide biased noisy labels are separated from reliable workers and the workers who provide useless labels.

(3) We propose an efficient method to recognize the biased sloppy workers. Due to the sparsity nature of crowdsourced data, we estimate the worker biases through calculating their expected error rate.

(4) We propose a truth discovery based approach to infer truths from both reliable workers and the corrected biased workers. Gold truth is not required to compute the estimated labels and worker reliabilities in our methodology.

To show the effectiveness of the proposed ITSC-TD framework, experimental results and analysis are presented on both synthetic and real world data sets. The results are compared with several prevalent comparison models, which include MV, TD, GLAD and EM-DS methods. The comparisons showed that our proposed ITSC-TD outperform other comparison approaches while the ratio of biased sloppy workers > 0.5, and significant improvement could be obtained in the simulated datasets. As a result, around 10% to 16% improvement for the accuracy were presented

in this case. For real world data, ITSC-TD is able to achieve better results regarding the selected measurement metrics while the proportion of highly biased workers > 0.5, comparing with multiple comparison methods. For example, comparing to comparison methods with TREC dataset, 2% and 8% improvement for accuracy, and 4% to 9% improvement for F measure can be obtained by using our proposed approach.

In the proposed approach, we assume there is no gold truth available while recognizing biased sloppy workers and inferring truths and worker reliability. It would be interesting to investigate the performance of the ITSC-TD method if a subset of the true labels are known. By utilizing these known truths, it might be possible to adjust the proposed method, such as the settings of prior distributions, for better inferencing of the unknown truths. Furthermore, efforts could be made to explore the possibility of utilizing the data from sloppy workers when there is a mixture of ordinal and continuous scale labels. The optimization based truth discovery framework utilized in our work makes it easier to extend the proposed ITSC-TD approach to mixed ordinal and continuous data types.

CHAPTER 5

WORK LIKE AN EXPERT: A MULTIPLE VIEWS APPROACH FOR CROWDSOURCING COMPLEX TASKS

Discussions about how to collect high quality labels with crowdsourcing has been presented by most of the recent research [112, 136, 137], since work performed by the crowd can greatly vary across individuals. In particular, a major drawback of most crowdsourcing services is that we do not have control over the quality of the crowd workers [50]. Chapter 3 provided an integrated framework for numerical labeling tasks which combines spam filtering and a bias detection algorithms, to obtain high quality consensus results. In Chapter 4, a new categorization of workers which remedy the overlapping problem in clustering crowd workers into groups in Chapter 3, has been proposed as a foundation for the development of the iterative self correcting methodology. This methodology mitigates the sloppiness, which resides in the crowdsourced data, by utilizing the highly biased sloppy workers. However, both Chapter 3 and Chapter 4 did not take into account the specialty of the challenge in complex labeling tasks through crowdsourcing. The typical tasks posted on popular crowdsourcing platforms such as Amazon MTurk are micro-tasks or HIT (Human Intelligence Task), which are low in complexity, independent and requires little time and cognitive effort to complete [12]. Complex tasks, however, usually poses a significant challenge: crowd workers in many cases need to possess knowledge or skills in specialized task domains [4]. In this chapter, discussions and analysis are presented about how to improve the quality of the estimated labels for complex labeling tasks. We investigate the workers$'$ strengths in different features of the task, and combine the framework developed in Chapter 4 to infer the truths for items which need to be labeled.

5.1 Labeling complex tasks through crowdsourcing

Crowdsourcing has become a powerful mechanism for accomplishing work via outsourcing the tasks to a large number of anonymous workers online. The strength of crowdsourcing is that it allows us to process the large scale of tasks that need human intelligence [138], such as translation, linguistic tagging, and visual interpretation. A wide range of applications (e.g. ESP Game, reCaptcha, and

Freebase [36]) have been developed on top of more than 70 crowdsourcing platforms such as Amazon Mechanical Turk (MTurk) and CloudCrowd. However, the works completed on micro-task markets like MTurk are primarily simple and independent tasks [12]. For example, the tasks might be labeling an image or judging relevance of a search result. Moreover, it is found that the quality of the data obtained in crowdsourcing is often lower than that of data collected in the ordinary controlled environment.

In contrast to the typical tasks posted on MTurk, which are low-in-complexity, self-contained, short, and requiring little specialized skill [12], much work in real world is usually more complex, interdependent and requires significant time and cognitive effort [139]. In addition, these complex and expert tasks, such as assessment of a research paper or a product design, in many cases requires the crowd workers possess knowledge or skills in specialized task domains [4]. However, due to the diverse background and education levels of the crowd workers, it is difficult to have workers which possess these specific skills. This leads to the problem of poor quality of the feedback from crowds even more outstanding. To combat this challenge, some crowdsourcing systems break down the complex task into smaller sub-tasks [54, 140] or provide expert rubrics to workers [4, 14], in order to achieve quality assurance and accomplish complex work by the crowds.

Decomposing a complex task into simpler subtasks, and requesting new solutions to subtasks from the crowd, is testified useful in improving the quality of the worker performance [12, 54, 140]. For example, an article writing task may ask one group of workers decide the scope of the article, another set to find and collect the relevant information, a third to write the narrative, and finally a fourth set to lay out the document and edit the final copy [12]. This type of crowdsourcing systems is more suitable for decomposable tasks, which are able to be decomposed into simpler subtasks. However, such task decomposition requires careful design and engineering of task-specific workflows. There is another type of complex tasks, such as in the domain of design critique [4, 90, 91] and academic/education learning activities (e.g. providing assessment to students′ performance on a homework/project) [6, 97, 141], is able to be accomplished better after defining expert rubrics for them. This type of task usually cannot be decomposed in a straightforward manner. Consider for example, there is a set of weather UI dashboard designs that need to be evaluated as a labeling task described in [4]. Figure 5.1 gives part of the UI designs used for labeling. If the task is decomposed into smaller subtasks, in this case, the UI design image is divided into several sub-images, and any label (e.g. is the UI giving too much information) given to sub-images cannot reflect the quality of the original UI. That means, in order to achieve desirable feedback for the UI design, it is necessary

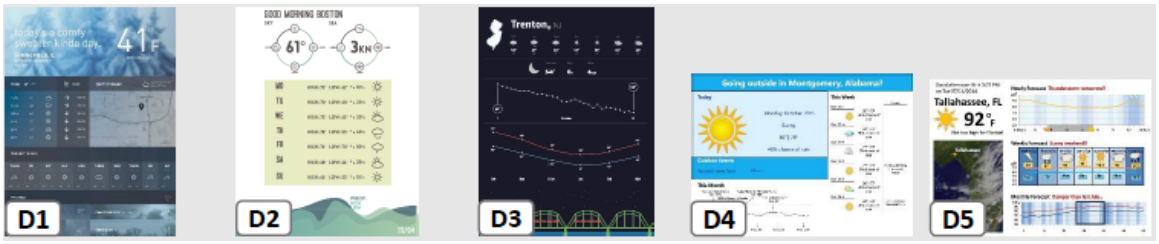


Figure 5.1: A subset of weather UI designs utilized in [4]

to review the item as a whole.

The research in this chapter concentrates on this category of in-decomposable complex tasks. As stated in [142], these complex tasks satisfy the following properties: (1) there is a large space of potential strategies that workers can use to solve the tasks, (2) workers have the capacity to solve the tasks by discovering and trying different strategies, and (3) a significant proportion of unskilled workers are unable to solve these tasks with high accuracy. As discussed before, a lot of recent research has been dedicated to improve the worker performance in completing the tasks by offering expert rubrics. The rubrics actually are the criteria or strategies which experts used to provide labels for the labeling task. Well-designed rubrics can help novice crowd workers align their own work with concrete labeling criteria, leading them to better grasp a domain's key principles [14,143,144]. Also, providing rubrics for workers would enable them to become aware of desirable work characteristics and goals. As a result, it leads to better quality labels achieved from the crowd workers.

Most of the prior work either demonstrate the plausibility of obtaining relevant and rapid feedback from the crowd, or the differences between expert and crowd workers. Very few of them investigate the workers' strengths or reliabilities on different evaluation aspects (rubrics). For example, while assessing a research paper written in English from a class student in the field of crowdsourcing. Some crowd workers (as reviewers) have the specialized knowledge in corresponding domain but might be influent in English. The other set of workers are native English speakers while having few backgrounds in crowdsourcing. It is possible that former group workers are more reliable than the latter ones when it comes to evaluating the technical parts of the paper. However, more credits might be given to the second group of workers, while giving an assessment of the presentation or writing aspects of the research paper. We investigate the characteristic of the crowd workers, and improve the quality of consensus results in a labeling task by utilizing the feature.

For easier reference, we utilize the "views" to denote the critiques or rubrics defined for the labeling task. It is inspired by the work in the domain of co-classification [145,146] and co-clustering

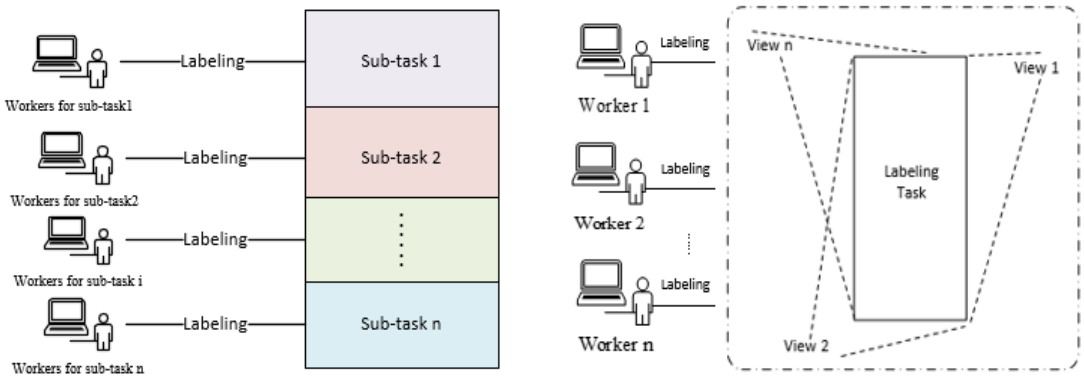
[147], which are two co-training frameworks for classification and clustering where independent views (e.g. distinct sets of attributes) of labeled and unlabeled data exist. Co-training is a semi-supervised learning algorithm, which first learns one separate classifier for each view using labeled data. The predictions on the unlabeled data, utilizing the learned classifiers, are then used as additional labeled training data [145]. In co-training frameworks, different views describe different kinds of information of an item. Similarly, we assume each view carries some information of the item which needs to be labeled, in crowdsourcing context. However, only the entire set of views defined by experts can be enough to describe the item, which is different from co-training where each view is assumed to be independent and sufficient for learning or describing the item. We give examples of the views defined in crowdsourcing setting and co-training to illustrate their differences:

- *Example of Views in Co-training*: While classifying a web page, two independent views can be defined: a) the text appeared on the document/web page itself, and b) the anchor text attached to hyper-links pointing to this page, from other pages on the web.

- *Example of Views in Crowdsourcing*: When giving an assessment to a research paper, a set of views can be defined: a) novelty of the article, b) adventurous of selected data sets, c) comprehensiveness of the paper, and d) enough examples and inferences.

To give a better understanding of the multiple views defined for a complex task, we give figure 5.2, with graphical illustration of the labeling process for a decomposable task and an in-decomposable task, respectively. Figure 5.2(a) presents labeling decomposable complex task by breaking it down into smaller subtasks, while Figure 5.2(b) shows the multiple views approach to obtain labels for the in-decomposable labeling task. It can be seen that in Figure 5.2(b), each view describes some information of an item, this view should be evaluated in the context of this task/item. As an example, if view1 is defined as novelty of a research article, the feedback to view1 should reflect the innovation of the entire paper, instead of just one section of it. After defining the views, it is possible to analyze ability on each view, and infer the true labels for the views based on the worker weights. Finally, combining the view labels to obtain a single result for each item, and investigating the quality of the estimated truths can be conducted. Also, the self-correcting truth discovery framework proposed in Chapter 4 is applied to filter the spam workers and correct the labels biased sloppy workers, for the purpose of achieving high quality results.

The rest of the chapter is organized as follows: Section 5.2 presents the related work in the area of improving the work quality for complex tasks in crowdsourcing systems. Section 5.3 presents



(a) Labeling for decomposable complex tasks

(b) Labeling for in-decomposable tasks

Figure 5.2: Illustration of labeling decomposable and in-decomposable tasks (In-decomposable tasks are labeled through multiple views)

the idea of investigating worker reliabilities on multiple views while estimating the true labels for items. Real world data sets are collected and described in Section 5.4 to evaluate the effectiveness of our multiple views approach. Section 5.5 presents the chapter summary.

5.2 Literature review on crowdsourcing complex labeling tasks

To achieve high quality feedback from crowd workers, especially for complex tasks, efforts have been made by many researchers recently. A set of research has attempted to crowdsourcing decomposable complex tasks by breaking them down into smaller subtasks that are easier to solve [6, 12, 13, 19, 54, 84, 148, 149]. The authors in [19] presented a framework to envision a future of crowd work that can support more complex, creative, and highly valued work. Figure 5.3 gives this proposed framework. The complex task is first decomposed into smaller subtasks, with each assigned to appropriate groups of workers, to collect output. Cheng et al. [148] explored the costs and benefits of decomposing macrotasks into microtasks for three task categories: receipt arithmetic, line sorting, and audio transcription. It is found that breaking these tasks into microtasks results in longer overall task completion times, but higher quality outcomes and a better experience that may be resilient to interruptions. Bernstein et al. [54] presented a word processing interface, which is called Soylent, to enable writers to call on MTurk workers to shorten, proofread, and otherwise edit parts of their documents on demand. They introduced the Find-Fix-Verify crowd programming pattern, which splits tasks into a series of generation and review stages, in order to improve worker quality. Kittur et al. [12] developed the CrowdForge framework and toolkit in their work for crowdsourcing decomposable complex tasks. Their approach builds on the general approach to simplified distributed

computing exemplified by systems such as MapReduce [87] of breaking down a complex problem into a sequence of simpler subtasks, using a small set of subtask types and managing the dependencies between them. The benefits and limitations of CrowdForge is demonstrated on case studies of article writing, decision making, and science journalism. The authors in [149] focused on crowdsourcing itinerary plans as a case study and introduced a collaborative planning system called Mobi, which takes as input a planning mission containing a set of constraints articulated by the user, and produces as output an itinerary that satisfies the mission. In this planning system, the idea of breaking down a task into subtasks to request solutions is implicitly utilized for better results. Noronha et al. [13] introduced PlateMate, a system for crowdsourcing nutritional analysis, such as calories, fat, carbohydrates, and protein, from photographs of meals using MTurk. To achieve accurate estimates, the authors propose a workflow in which the overall problem is decomposed into small, manageable, and verifiable steps. VanHoudnos [6] presented crowdsourcing as a novel approach to solve the problem that a local charter school faces with the scoring of a constructed response exam. The expert-tasks are split into small grain-size work to collect ratings on MTurk. Zhang et al. [84] investigated the interplay between algorithmic paradigms and human abilities and interests for the purpose of crowdsourcing general computation that enables the solution of new classes of tasks via human computation. To enable effective and efficient coordination among human problems solvers, algorithmic paradigms such as divide-and-conquer for decomposing a problem into sub-problems, and for composing solutions of sub-problems into solutions are drawn in [84]. Lasecki et al. [150] presented an end-to-end system LEGION:SCRIBE, to allow deaf people to request captions at any time. The system breaks down audio transcription tasks into small chunks and uses sequences alignment to combine results in real-time. There were also some toolkits seek to support decomposable complex applications developed in literature. TurKit [151] aids requesters in deploying iterative tasks on MTurk. Its crash-and-rerun architecture saves intermediate results to avoid redundant crowd assignments. Jabberwocky [152] can target workers in both paid and volunteer workforces. In these toolkits, it is assumed that task designers will determine how tasks are broken down in all cases [140].

Decomposable tasks usually requires careful design and engineering of task-specific workflows [142]. There are a set of complex labeling tasks, which are difficult to accomplish task decomposition —break down a task into smaller sub-tasks. These in-decomposable tasks often need to be solved within the global context. As stated in [149], "Within studies of human computation, an important class of underexplored tasks is those in which the solution must satisfy a set of global requirements".

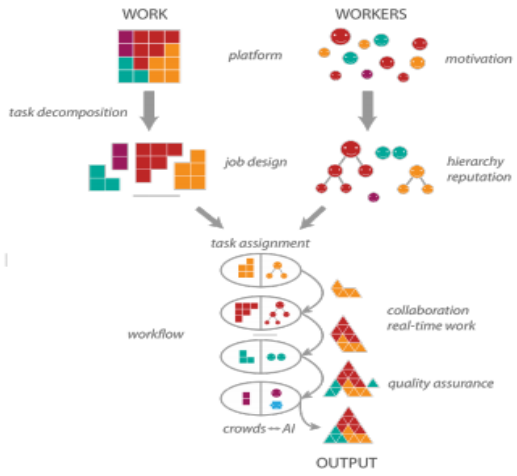


Figure 5.3: Proposed framework to support complex and independent crowd work in [19]

For example, while evaluating an essay, it is required to present a balanced perspective upon different components that comprises the article. In particular, creative tasks such as graphic design or assessment will need to be solved by relying on the composition as a whole and are marked by interdependence among solution components. Such tasks are usually not amenable to the divide and conquer approach. There is some research that explored how to tackle these indecomposable complex tasks via crowdsourcing [4, 14, 91, 142, 153–156]. Several prior works focus on the training of crowd workers to solve tasks with high quality feedback [153–156]. Oleson et al. [153] used gold standards as a form of training in their work, for quality assurance, to prevent common scamming scenarios. Willet et al. [154] proposed to use examples for training workers and for calibrating their work, to match the task requesters' expectations on visualization tasks. They found that workers exposed to examples generated higher quality responses than workers who did not. Dontcheva et al. [155] proposed a platform that integrates the training and crowdsourcing in a photo editing environment. Workers gain new skills through interactive step-by-step tutorials and test their knowledge by improving real-world images submitted by requesters. Singla et al. [156] developed a model to select examples to teach workers in classification tasks. Mitra et al. [157] used a qualification test to screen workers in qualitative coding techniques, and found that this training is quite effective in improving the quality of the data annotations.

The problem with training crowds for performing the complex tasks is that a set of gold standards or examples is required to be used to screen the workers. In many cases, this is not always available. In addition, most gold standards sets used in the training process are just simple micro-tasks, which are for the purpose of preventing common scamming cases, so the workers who

qualified through the training sets do not necessarily perform well on actual complex tasks.

Another set of research focuses on improving worker performance in crowdsourcing in-decomposable complex tasks on the basis of learning sciences [4, 14, 158, 159]. As indicated in [142], in learning sciences, "instructional interventions have been more intensively studied than in the crowdsourcing community". Studies have tested and shown worked examples (or expert solutions) are effective in skill acquisition [158, 159]. The research in [159] and [160] presented that reviewing worked examples is more effective than solving the task in the learning process, especially for novices. Providing worked examples to crowd workers for learning is useful in less-structured tasks, where domain skills can be learned from step-wise expert solutions. However, it is sometimes difficult to obtain the required knowledge from tasks such as design critique. Instead of offering worked examples, prior work has demonstrated the effectiveness in providing rubrics, which denote the strategies or principles experts used in solving the complex tasks [4, 14, 97]. By providing rubrics such as scoring templates, workers will be made aware of the desirable work characteristics, and can learn by aligning the characteristics with their own work [144]. Dow et al. [14] designed the Shepherd system to manage workflows for tasks posted on MTurk. Their work showed that crowds can be shepherd via offering expert rubrics, and utilizing task-specific rubric yields better results. Yuan et al. [4] evaluated the use of expert rubrics in helping crowd workers provide high quality feedback, with crowdsourcing the task of assessing weather UI dashboard designs. The work focused on the salient differences between experts and novice workers' feedback. Saddler and Good [97] suggested in their study that with the help of rubrics (provided to students), there is a high correlation between students and their teacher on test questions. The work in [90–92] applied rubrics to help crowd workers to provide rapid and relevant feedback, in the area of design critique. Kulkarni et al. [161] found in their research that well-designed rubrics can help lower the variance and improve accuracy in the grading process, with peer and self assessment, for massive online classes. Luther et al. [91] presented the CrowdCrit crowdsourcing system, in which workers use rubrics of design principles to give critique statements for UIs created in design contests.

Most of the prior work demonstrates the effectiveness of obtaining relevant and rapid feedback from crowd workers by offering rubrics for them, or gives analysis of the differences between the experts' solutions and crowd answers. However, they do not take into account the fact that workers' reliabilities may vary on different views of the task. Each view described a certain attribute or principle (rubric) of a task, in specialized task domain. Our work in this chapter concentrates on crowdsourcing the in-decomposable complex scoring tasks, which represents the labeling tasks with

TABLE 5.1

List of principle statements that comprise the rubric [4]

Principle Statement	Principle Description
Need to consider audience	The design does not fully consider the target users and the information that could affect their weather-related decisions.
Provide better visual focus	The design lacks a single clear 'point of entry', a visual feature that stands out above all others.
Too much information	Take inventory of the available data and choose to display information that supports the goals of this visual dashboard.
Create a more sensible layout	Information should be placed consistently and organized along a grid to create a sensible layout.
Personalize the dashboard	The design should contain elements that pertain to the particular city, including the name of the city.
Use complementary visuals and text	The design should give viewers an overall visual feel and allow them to learn information from text and graphics.
Needs a clear visual hierarchy	The design should enable a progressive discovery of meaning. There should be layers of importance, where less important information receives less visual prominence.
Thoughtfully choose the typeface and colors	The type and color choices should complement each other and create a consistent theme for the given city.
Other	Freeform critique that does not fit into the other categories.

ordinal or numerical scales. We investigate the trustworthiness, which defines the worker's reliabilities, on a set of views defined for the task, and apply weighted voting to each view to improve the consensus results, for the items which need to be labeled.

5.3 Crowdsourcing complex scoring tasks – a multiple views approach

In this section, we present the idea of multiple views approach to obtain high-quality labels for complex tasks through crowdsourcing. Although defining views is a way to help workers grasp the goal and domain knowledge of the task, spamming and biasing scenarios are still inevitable. To filter spam workers and biased sloppy workers among the crowd, the iterative self correcting – truth discovery (ITSC-TD) framework proposed in Chapter 4 is applied to achieve better consensus results. Section 5.3.1 presents the introduction of the multiple views approach for labeling tasks through crowdsourcing. The ITSC-TD combined with multiple views method is presented in Section 5.3.2. Some of the concepts used are given as follows:

Complex Task – The task which requires workers possess knowledge or skills in task domain. The complex tasks from this point forward are referred as in-decomposable tasks by default.

Complex Scoring Task – It indicates the complex labeling task with categorical ordinal or numerical labels. The work in this chapter focuses on complex scoring tasks.

View – Each view carries some information such as principle or sub-goal of a task. It reflects a subset of desirable work characteristics for the task. It is defined by the expert, and provided to crowd workers in order to help them complete the labeling task. Similar to rubric, it can help

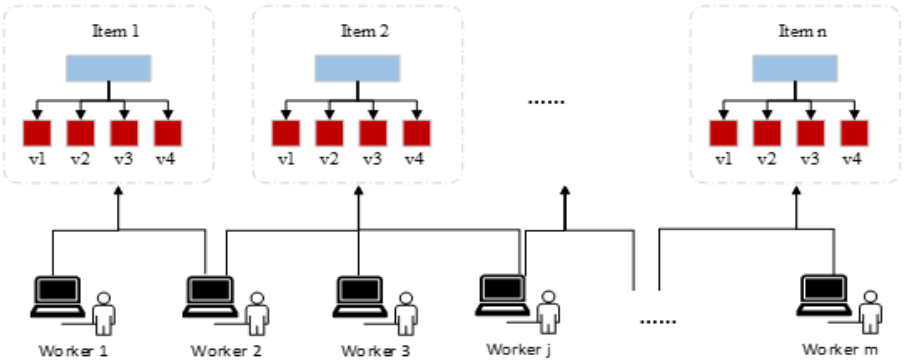


Figure 5.4: Crowdsourcing complex scoring task with multiple views provided

workers align their work with the labeling criteria and goals, and lead them to have a better understanding of the domain principles [14, 143, 144].

5.3.1 Worker reliabilities on task views – multiple views approach

The study in this chapter explores how expert defined views affect the answers provided by the crowd workers. It further seeks to improve the quality of the consensus labels by analyzing worker performance on each view. The research questions explored are: 1) Are the worker reliabilities consistent or varying on different views for a scoring task? 2) Is it possible to improve the final estimated labels through investigating task views respectively?

In general, a set of views are defined by experts for a labeling task, since experts usually own the required domain skills and knowledge to accomplish the task, and their labels are often the closest answers to gold truths. The view set should be able to cover the domain key principles in order to accomplish the task. For example, Yuan et al. [4] gave a set of principle statements in the task of evaluating weather UI dashboard designs. Figure 5.1 showed some of the designed UIs. In order to obtain assessments for the designs, they recruited crowd workers from some popular crowdsourcing platforms, and offered the principle list, which is presented in Table 5.1, to the workers. Each principle could be seen as a view as defined in our work. The principle list gives guidance to the workers about the characteristics or strategies experts utilized to assess the UI designs. The crowdsourcing process for labeling task with multiple views defined is shown in Figure 5.4. As shown in Figure 5.4, each worker gives feedback to the views defined for the items in a labeling task. Once the view labels are obtained, they can be combined to get a single label for an item.

Assume weighted voting is utilized to obtain the consensus labels for views of each item.

There are two different ways to achieve the results:

(1) ***Same weights on all of the views (Single weight for each worker)***: In this approach, a worker's reliabilities on different views are the same. In other words, each worker has a single weight, and the weight is decided through minimizing the overall deviations between observed labels and true labels on all views. To give a detailed explanation, we show the process of determining the worker weights, using the optimization based truth discovery framework [18, 132], which is utilized for estimating the consensus labels in our work.

Assume M views are defined for labeling the items in a task. K workers have participated in the labeling task, and there are total N items need to be labeled. Let $v_{im}^{(k)}$ denotes the label for view m of item i, given by worker k, and $v_{im}^{(*)}$ as the truth of the m^{th} view for the i^{th} item. Worker weights are denoted as $W = \{w_1, w_2, ..., w_k\}$. $Y^{(*)}$ is the truth table which stores the true labels for each view of the items, and $v_{im}^{(*)}$ is an element of the truth table. In the crowd scoring task, we assume that the possible labels are always > 0, and $v_{im}^{k} = -1$ if a worker does not give a label on the view of item i. The optimization framework that estimate truths is as:

$$\min_{\{w_j\},\{v_{im}^{(*)}\}} f(Y^{(*)}, W) = \sum_{k=1}^{K} w_k \sum_{i=1}^{N} \sum_{m=1}^{M} (d_m(v_{im}^{(*)}, v_{im}^{(k)})) \cdot (1 - \mathbb{1}\{v_{im}^{(k)} = -1\}) \\ s.t. \quad \delta(W) = 1, \quad W \in S \tag{5.1}$$

where d_m is the loss function defined for the m^{th} view. It measures the distance between the true label $v_{im}^{(*)}$ and the observation for the k^{th} view $v_{im}^{(k)}$. $\mathbb{1}(\cdot)$ is the indicator function. $\delta(W)$ is the regularization function, which reflects the distribution of the worker weights in domain S. The most widely used regularization function [128, 132] is as follows:

$$\delta(W) = \sum_{k=1}^{K} exp(-w_k) \tag{5.2}$$

From Eq(5.1), it can be found that the weight of each worker is determined through minimizing the summation of the errors made on all the views. Thus, only one weight is assigned to each worker. It is equivalent to have same weights on all the views for a certain worker.

(2) ***Varied weights on different views (proposed multiple views approach)***: it analyzes the worker's weight on each view separately. Compared to the approach in (1), this method is especially useful for the following scenario: Assume two workers w_1 and w_2 worked on two different items in a labeling task, 2 views are defined as $\{v1, v2\}$. Let the distance between

truths and observations on the views be $[0,1]$ on item1, $[0,5]$ on item2 for w_1, and $[1,1]$ on item1, $[1,1]$ on item2 for w_2. The total error for $w_1 = 6$, and $w_2 = 4$. If single weight is assigned to each worker based on the overall errors/deviations which the worker made, then in this case, w_2 should be trusted more compared to w_1, for both v_1 and v_2. However, it is clear that actually w_1 is more reliable on v_1 compared to w_2. While computing the consensus labels for v_1, it would be more reasonable to assign higher weights to w_1, instead of w_2. Thus we propose to give weights based on the worker performance on each view, and investigate how it affects the final consensus results.

Similar to single weight per worker in approach (1), optimization based truth discovery [18, 132] is utilized to estimate the truths for each view. Assume $w_k^{(m)}$ is the weight of worker k on view m, the process to estimate true labels for v_m (view m) is:

$$\min_{\{w_j^{(m)}\},\{v_{im}^{(*)}\}} f(Y_m^{(*)}, W^{(m)}) = \sum_{i=1}^{N}\sum_{k=1}^{K} w_k^{(m)} \cdot d_m(v_{im}^{(*)}, v_{im}^{(k)}) \cdot (1 - \mathbb{1}\{v_{im}^k = -1\})$$
$$s.t. \quad \delta(W^{(m)}) = 1, \quad W^{(m)} \in S \tag{5.3}$$

The two discussed approaches present the ways to assign weights to workers, and estimate the true labels for each view, which is called estimated or consensus view labels. The estimated view labels are then combined to get a single value/label for each item. How to combine the consensus view labels depends on specific applications. For example, if a task is to grade students' submissions on a class project, and the expert split the overall grade into a set of views. Then adding up the estimated view labels as the approach to combine them into an estimated label for each submission would be appropriate. Otherwise, some regression or classification algorithms, such as linear regression, logistic regression, or Naive Bayes, could be used to obtain the item labels, from the consensus view labels.

5.3.2 Multiple views approach together with ITSC-TD framework

In order to remove the labels which are useless to infer truths, and correct the feedbacks obtained from biased workers, we utilize the ITSC-TD framework proposed in Chapter 4 on each view, to improve the quality of the consensus view labels. Figure 5.5 gives the overview of the framework proposed which combines multiple views with the ITSC-TD method. First of all, a set of views is defined by the expert to guide the workers to align their work with the labeling criteria. The workers' reliabilities are analyzed on each view within the view set, and they are

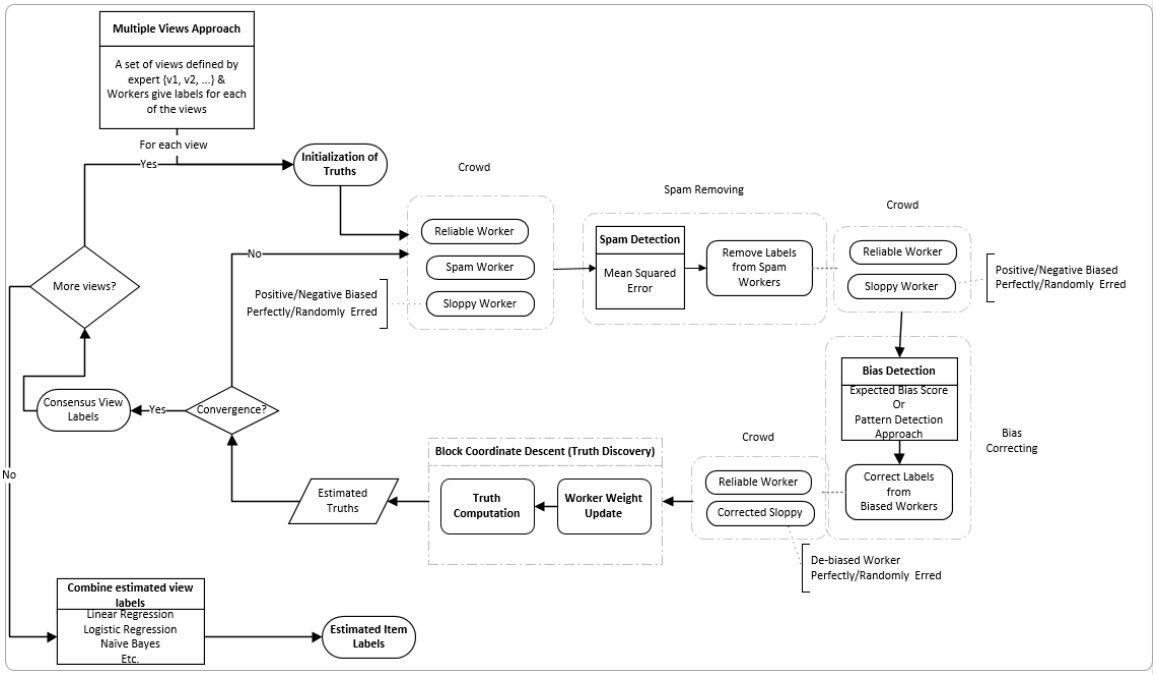


Figure 5.5: Multiple views approach with ITSC-TD framework

utilized to weight the observed view labels to get the consensus view labels. The worker reliability and view label estimation is based on the optimization based truth discovery framework proposed in Section 4.3.3. One great advantage of utilizing this framework is that it could be used for both categorical and numerical labeling tasks. In the process of estimating the true labels for each view, the spam workers and biased sloppy workers on this specific view are tackled respectively: The spam workers are filtered out from the crowd workers, and the labels provided by biased sloppy workers are corrected according to their bias types (either positive or negative biased types). Finally, the consensus view labels are combined to obtain the estimated truths for items.

The proposed ITSC-TD framework in Chapter 4 is utilized for tasks with categorical ordinal labels. So in order to adapt the methodology to numerical data, we utilize the pattern detection approach developed in Chapter 3, Section 3.2, to recognize the positive or negative bias patterns for the biased workers on each view, and correct their labels based on the detected patterns. For spam filtering on the labeling task with numerical scale for a specific view, we discretize the observed labels by mapping them into an ordinal scale. Spam workers are then removed based on Eq(4.29). The mapped ordinal scale usually is defined by an expert, it specifies the error tolerance range of an application. As an example, if a grading process is considered, where the label scale is numerical data from 0 to 100, then the observed numerical labels could be mapped into ordinal scale $\{1, 2, 3, 4, 5\}$.

This ordinal scale is equivalent to grade scale which is mostly used in many education systems: $\{F, D, C, B, A\}$.

For the loss function, which is indicated as $d_m(\cdot)$ in Eq(5.1) and Eq(5.3), 0-1 loss is used for ordinal labeling tasks. As for the continuous data, normalized squared loss is applied to characterize the distance between observed view labels and the truths. Normalized squared loss is a common used loss function [132], which is defined as:

$$d_m(v_{im}^{(*)}, v_{im}^{(k)}) = \frac{(v_{im}^{(*)} - v_{im}^{(k)})^2}{std(v_{im}^{(1)}, ..., v_{im}^{(K)})} \tag{5.4}$$

Assume worker 1 to K gives labels on view m of item i. According to Eq(5.3) and Eq(5.4), the estimated view truth which minimizes the weighted distance on a specific view should be:

$$v_{im}^{(*)} \leftarrow \frac{\sum_{k=1}^{K} w_k^{(m)} \cdot v_{im}^{(k)}}{\sum_{k=1}^{K} w_k^{(m)}} \tag{5.5}$$

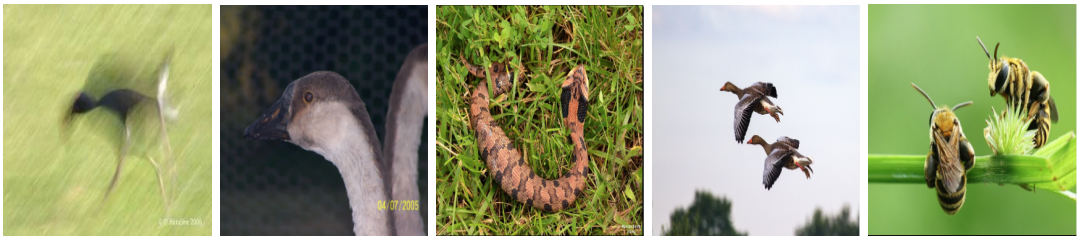
The obtained $v_{im}^{(*)}$ is the estimated view label for an item. It is collected and used to feed into the combination algorithm to obtain a single estimated label for each item.

5.4 Experimental results and analysis

In this section, the multiple views approach combined with ITSC-TD framework is evaluated on two collected real world data sets. Worker reliabilities on different views are investigated, by comparing the experimental results of assigning single weight to each worker to the approach with varied weights on views. Section 5.4.1 presents the collected real world data sets for evaluating the effectiveness of the proposed multiple views approach. Section 5.4.2 presents the evaluation metrics utilized for measuring the output of the experiments. Impact of multiple views approach on the estimated true labels, and the results analysis is presented in Section 5.4.3.

5.4.1 Real world data sets

To evaluate the proposed multiple views approach, which investigates worker reliabilities on different views, we collected two sets of real world crowdsourced data, for conducting the experimental analysis. A set of views are defined for both of these two labeling tasks, and crowd workers are then requested to provide feedback for the tasks. The description of the data sets is presented as next.



(a) (b) (c) (d) (e)

Figure 5.6: Examples of animal images used for labeling task through Amazon MTurk

(1) *Image Labeling dataset*: A set of images are used to collect labels through crowdsourcing. The data set – "How beautiful is this image (Animals)", which includes an URL for each image, is downloaded from CrowdFlower[1] Data for Everyone and utilized for requesting labels. The task is to give a rating for image quality on the scale of 1-5. The original dataset contains 3601 urls of images which contain animals, we randomly select 500 out of them to collect labels from Amazon Mechanical Turk[2]. Figure 5.6 gives examples of images we selected for the labeling task.

Five different views are defined for the image quality rating task, and they are illustrated as follows:

(a) How do you rate the resolution of the image?

(b) Is this image useful (e.g. Could the image be used for magazine, book, etc.)?

(c) How do you like the animal in the image?

(d) How do you like the presentation of the animal image (e.g. Could the animal be fully seen in the image)?

(e) How do you like the scene presented in the image as a whole?

The rating scales on the views are also from 1 to 5. Two different sets of labels are collected from the crowds. (I) Labels for image qualities without views provided for workers. Each worker provides a label for the overall quality of the image, without exposure to the defined views. The question being answered by the crowd is: Overall speaking, how beautiful is the image? A rating is selected from a scale from 1 to 5. (II) Each worker offers labels for all the defined views on each image. An image together with the defined 5 views are presented to the workers,

[1]https://www.figure-eight.com/data-for-everyone/
[2]https://www.mturk.com/

TABLE 5.2

Views defined for project grading task

View Statement	View Description
Data set selection	Selection of data set for data mining - how complex, how large, how challenging, is it real world data?
Understanding and novelty of the project	Does the report gives good explanation of the problem and the goals of the data mining project? Is the problem real world or repetition from Internet?
Use data preprocessing	Does the report gives explanation of preprocessing methods used, and discussion of results obtained? How serious is preprocessing (how noisy is real world data)?
Proper data mining techniques	Data mining techniques used in analysis should be minimum 3 or more. Explanation of tool use and tuning the parameters of algorithms used need to be presented. How's the presentation of obtained results?
Gives discussion of results	Are all presented diagrams and tables explained and discussed with enough details? Comparisons of methodologies (ROC analysis or similar) should be given.

and evaluations on scale 1-5 are requested for the views from the crowd. In this process, every worker will have to give ratings for all 5 views of an image before proceeding to next one.

For both (I) and (II), each image is requested to be labeled by 6 workers. The reward per HIT we offered on MTurk platform is \$0.05. In order to evaluate the experimental results, we hand labeled all these 500 images to serve as the gold truths. As an example, the true labels we given for images (not views) from (a) to (e) in Figure 5.6 are corresponding to 1 to 5. Finally, 68 workers are recruited to complete the labeling tasks in (I) and (II) respectively, and 3000 labels are obtained for case (I) and 3000 sets of view labels for (II).

However, in the label set (II), there are 11 workers only provide 1 set of view labels per worker. In other words, each of the 11 workers only rated 1 image per person. We removed these workers and their labels, by preprocessing the collected data set, since there is not enough data to estimate the worker reliability in this scenario.

(2) *Project Grading dataset*: We collect students' project reports from the graduate-level course – data mining class (CECS 632) offered in University of Louisville, and obtain evaluations (grades) for these reports from the crowd. Reports for two project assignments are collected: Project 1 and Project 2, with 11 students' submissions for Project 1, and 12 submissions for Project 2. For both of the two projects, each student submitted a report which concentrated on solving a data mining task, with techniques such as classification, regression, clustering, or outlier detection, etc. Students were able to choose their own data sets of interest to perform experiments and analysis.

Different from the image labeling task, grading the data mining project reports requires more efforts and specialized domain knowledge. In this case, the workers who are supposed to

TABLE 5.3

Some statistics about the collected real world datasets

Data Set	Label Type	Label Scale	View Scale	View Num	Instances	Workers (I)	Workers (II)	Ratings (I)	Ratings (II)
Image Quality Labeling	Ordinal	5	5	5	500	68	57	3000	2989
Project Grading	Numerical	20	4	5	23	25	25	100	100

I: Represents the label sets obtained without providing views to crowd workers;
II: Represents the label sets obtained using defined expert views;
Label Scale: The scale used for the rating on item as a whole (not on views).

provide evaluations for project submissions would need to at least understand basic concepts in the field of data mining. Thus, instead of crowdsourcing the tasks through MTurk, we recruited workers/graders within the Computer Engineering and Computer Science (CECS) department from University of Louisville. As a result, the recruited crowd workers includes both undergraduate and graduate students with a background in data mining area.

Similar to the image labeling data, two sets of labels are collected from the workers: (I) Grades for the project reports based on the understanding of workers, without any critique principles (views) provided. The rating is on the scale of 20 (numerical data). (II) View grades for items. The views given for the task are the assessment principles, and they are defined by the professor of the data mining course. In the grading process, the reports together with the view sets are provided to workers, to request view grades from the crowds. Table 5.2 presents the 5 views defined for the evaluation task. The max point for each view is 4. The total point for the whole report would be out of 20 ($4 * 5$), which is consistent with the label scale in (I). Both of these two sets of data involve 25 workers/graders.

Each report is at least graded by 4 workers. There are 100 grades in total obtained for the task in (I), and 100 sets of view labels achieved for (II). The grades given by the advisor of the class are collected as the true labels for the project reports, and they are used to evaluate our proposed framework. Table 5.3 gives the statistics about the two real world datasets collected in our work.

5.4.2 Evaluation metrics

Both ordinal and numerical data is used to conduct the experimental analysis in our work. In order to measure the performance of the proposed multiple views approach, we utilize the following evaluation metrics:

TABLE 5.4

Impact of expert defined views on the estimated true labels for items

Data Set		no views		views	
		MV/AVG	TD	MV/AVG	TD
Image Quality Labeling	Accuracy	0.19	0.214	0.396	0.436
	F1	0.167	0.206	0.344	0.410
Project Grading	σ	2.893	2.140	1.760	1.244
	ρ	0.682	0.701	0.765	0.816
	$RMSE$	2.950	2.351	2.040	1.549

TD: Optimization based truth discovery framework
MV is applied to image quality labeling data set
AVG is applied to project grading data set

- *Labeling task with ordinal labeling scale*: Accuracy and F1 measure will be adopted as the measurement of the effectiveness of the proposed framework.

- *Labeling task with numerical labeling scale*: For numerical data, standard deviation (σ), correlation coefficient (ρ), and Root Mean Square Error ($RMSE$) are used as performance measurements for the experimental output.

5.4.3 Experiments of multiple views approach on real world datasets

Before evaluating the proposed multiple views approach, we first validate the idea that it is possible to improve the work performance of the crowds, by providing views. The optimization based truth discovery framework is applied to estimate the true labels for label sets with and without views. We do not tackle the spamming and biasing problems in this process, since it is only of interest to investigate the impact of providing expert views, which are labeling criteria/principles, to the crowds.

(I) Impact of providing expert views for the crowd

The item label sets, which obtained without information of views, are first used to achieve consensus labels through (a) Majority Voting (MV) for ordinal data, or Averaging (AVG) the labels for numerical data, and (b) the optimization based truth discovery framework as indicated in Eq(4.22). The results are compared to the estimated truths for the items with the expert view provided to crowd workers. To obtain the single estimated label for each item from the observed view labels set, we proceed the truth discovery process as follows:

(1) Utilizing aggregation algorithm to get the consensus labels for the views of each item. Two different approaches are testified and compared: a) Majority Voting (MV), and b) the approach denoted in Eq(5.1) and Eq(5.2).

TABLE 5.5

Significance test – Tukey HSD results

Data set		Models without views v.s. Models with views	
		F	p
Image Quality Labeling	Accuracy	84.1838	0.0117
	F1	24.6998	0.0382
Project Grading	σ	4.9406	0.1563
	ρ	13.2357	0.0679
	$RMSE$	4.8859	0.1577

(2) Combine the consensus view labels to give the final single estimated true label for each item. For the collected project grading dataset, the points for a whole project report have been split into each view by an expert, thus estimated truth for each report could be obtained by adding up the consensus views labels of the item. However, for the image labeling dataset, it is not applicable to just simply sum the views labels to get the overall rating for every image. In this case, two different models – Naive Bayes (NB) and Softmax Regression (multinomial logistic regression), are built to map the consensus views labels into overall image labels. The models are developed by utilizing a subset of gold truths, which are the overall labels and corresponding views labels for each image offered by experts. In particular, the features of the model are the labels for the views of an item, and the dependent variable is the rating for the item as a whole. The experimental results showed that softmax regression gives better results than NB. Thus, we only present the output for softmax regression model for analysis.

To build the softmax regression model for the image labeling data, 100 items (images) are selected randomly from the gold truth set for the training process. Table 5.4 gives experimental results for the two collected datasets. The best output can be obtained by defining views, and utilizing optimization based truth discovery method for both image quality rating dataset and project grading dataset. To analyze the impact of the defined views on the work performance, we conduct significance test on the calculated evaluation metrics. One way ANOVA with post-hoc Tukey HSD test is calculated for the analysis.

Table 5.5 gives the significance test results comparing the models with views and approaches without views. If we set the significance level as 0.05, then based on the calculated p-value, it could be found that significant improvement can be obtained for both accuracy and F1 measures, for the image rating dataset. For the project grading dataset, although p-values for standard deviation, correlation coefficient and $RMSE$ are > 0.05, there is still great performance improvement for

TABLE 5.6

Experimental results of single weighted & varied weights on different views models

Data Set		Single Weighted Model (TD)						Varied Weights Model (TD)					
		v_1	v_2	v_3	v_4	v_5	Overall	v_1	v_2	v_3	v_4	v_5	Overall
Image Quality Labeling	Accuracy	0.336	**0.328**	0.228	**0.352**	**0.344**	0.436	0.342	**0.430**	0.228	**0.464**	**0.435**	0.540
	F1	0.307	**0.311**	0.207	**0.351**	**0.341**	0.410	0.326	**0.415**	0.207	**0.463**	**0.408**	0.537
Project Grading	σ	0.510	0.475	**0.695**	0.720	**0.642**	1.244	0.500	0.473	**0.574**	0.720	**0.544**	1.031
	ρ	0.786	0.803	**0.685**	0.673	**0.727**	0.816	0.788	0.805	**0.751**	0.673	**0.770**	0.892
	$RMSE$	0.617	0.583	**0.716**	0.809	**0.702**	1.549	0.612	0.583	**0.637**	0.809	**0.629**	1.385

these evaluation metrics. For example, through defining views, we could reduce the RMS error by around 30.8% with AVG as the aggregating method, and 34.1% with the optimization based truth discovery approach. It can be concluded that expert defined views have a positive effect on workers' performance in complex labeling tasks, in many cases.

(II) Experimental results on multiple views approach

In order to investigate the impact of taking into account workers' reliabilities on different views, we conduct experiments with single weight for each worker, as well as the varied weights on the view set. As shown in Table 5.4, optimization based truth discovery (TD) gives better results compared to MV or AVG algorithms, so we only show the outcome of the TD method for experiments on multiple views approach. Here, we do not apply the spam filtering and bias correction algorithms. The experimental outputs of single weight per worker model and varied weights approach are presented in Table 5.6.

In Table 5.6, the highlighted cells represents the views which achieved improvement after utilizing the multiple views approach with varied weights assigned to different defined views. The image quality labeling data set showed view 2, 4 and 5 obtained higher accuracy using the varied weights model, which is the proposed multiple views approach, compared to the single weighted method, while the project grading data set indicated view 3 and 5 have better results with the varied weights model. If we conduct significance test on view 3, 4, and 5 of the image labeling label set, the Tukey HSD p-value is calculated as 0.006 for accuracy and 0.011 for F1 measure (assume significance level is 0.05), so we could conclude that there are significant improvements on these views after applying different weights on views set. The overall column in Table 5.6 shows the quality measurements for the estimated truths for items, after combining view labels into a rating for every item as a whole. By comparing the results for the two models, 23.9% improvement could be obtained for accuracy, and 31.0% for F1 measure, on image labeling data set.

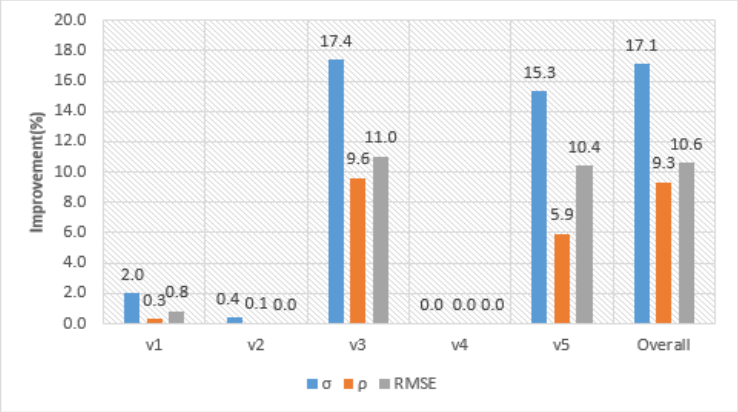


Figure 5.7: Percentage of improvement for the evaluation metrics on project grading data set

Although for project labeling data set, no significant difference can be obtained, we could still improve the results by reducing the σ up to 17.1%, and RES error up to 10.6%, on the final estimated overall labels. Figure 5.7 presents the percentage of improvement on each view, and the overall performance on the project grading task. The reason why no significant improvement can be seen on project grading data set, even after providing views for workers or using varied weights on different views, could be that the graders we recruited for completing the grading task are not like the novice workers in micro-task crowdsourcing platforms such as MTurk. These graders have already had education background in the specialized task domains. They might even have the experience in working on similar grading tasks before. For example, some of the graders, as we know, are PhD students who have been teaching assistants on data science related courses. In other words, some of the recruited workers could give grades very close to experts' ratings, even without providing enough information about the labeling principles.

It can be seen from Table 5.6 that evaluation metrics on view 1 and view 3 for image labeling data set, and view 1, 2, and 4 for project grading set, are almost the same on both models. The reason for the little change comparing varied weights model to single weighted model is that the rank of worker reliabilities obtained on these views, while applying multiple views approach, are consistent with the weights calculated using the single weighted method.

(III) Combining ITSC-TD with multiple views – integrated multiple views approach

In this section, we integrate the Iterative Self-Correcting Truth Discovery (ITSC-TD) with the multiple views approach. The ITSC-TD model removes the labels provided by spam workers, and correcting the biased sloppy workers' answers while estimating the truths from the observed

TABLE 5.7

Number of different types of workers detected using gold truths

Worker Types	Image Quality Labeling					Project Grading				
	v_1	v_2	v_3	v_4	v_5	v_1	v_2	v_3	v_4	v_5
Reliable	30	33	27	35	33	16	18	15	13	15
Spam	11	7	13	8	10	2	3	3	5	4
Biased Sloppy	6	8	2	4	3	4	1	2	3	2
Randomly Erred Sloppy	10	9	16	10	11	3	3	5	4	4

feedbacks from crowds. For each view within the views set, ITSC-TD framework is applied to obtain consensus view labels. The estimated labels for views are then combined to achieve an overall label for each item. The entire procedure is presented in Figure 5.5.

Table 5.7 gives the actual total number of spam workers and biased sloppy workers for each view for the two data sets, by comparing to the gold truths. The randomly erred sloppy row denotes the sloppy workers which do not have recognized bias patterns. This statistics about the data sets justifies the idea of proposed multiple views approach, which assigns varied weights on different views to each worker. As indicated in Table 5.7, different number of crowd workers are detected on different views, under the same worker types on a data set. For example, the number of spam workers on view 1 (6) is different from view 3 (2) the for image labeling data set. It gives the information that worker reliabilities are inconsistent on the views set. Therefore it would be useful to analyze each worker's performance per view.

The results for integrating multiple views approach with ITSC-TD framework are presented in Table 5.8. In order to filter the spam workers for each view in project grading data set, the numerical observed view labels are mapped into ordinal label scale $\{1, 2, 3, 4, 5\}$. The threshold for *RandomSep*, which is shown in Eq(4.29), is set to give the best results for each data set. The workers with calculated *RandomSep* greater than the threshold are removed as spammers. Similar to Chapter 4, the biased sloppy workers are detected using expected bias score for image labeling data set. For project grading label set, we recognize worker's bias pattern through the approach in Section 3.2, Chapter 3. The labels provided by biased workers are corrected by subtracting (positive bias pattern) or adding (negative bias pattern) the average of deviations between observed view labels and estimated view labels here.

As shown in Table 5.8, after applying worker filtering approach (ITSC) in the process of truth discovery, which estimates the true label for each view of an item, the quality of consensus view labels could be greatly improved. As a result, better overall estimated item label, which is the

TABLE 5.8

Experimental results for multiple views approach (varied weights model) & integrated multiple views (with ITSC-TD framework)

Data Set		Multiple Views (TD)						Multiple Views & ITSC-TD					
		v_1	v_2	v_3	v_4	v_5	Overall	v_1	v_2	v_3	v_4	v_5	Overall
Image Quality Labeling	Accuracy	0.342	0.430	0.228	0.464	0.435	0.540	0.498	0.584	0.350	0.605	0.584	**0.691**
	F1	0.326	0.415	0.207	0.463	0.408	0.537	0.487	0.573	0.328	0.597	0.579	**0.682**
	Detected Spam	-	-	-	-	-	-	6	4	6	5	5	-
	Detected Biased	-	-	-	-	-	-	3	3	0	2	1	-
Project Grading	σ	0.500	0.473	0.574	0.720	0.544	1.301	0.428	0.431	0.513	0.634	0.495	**1.059**
	ρ	0.788	0.805	0.751	0.673	0.770	0.892	0.895	0.890	0.796	0.702	0.801	**0.916**
	$RMSE$	0.612	0.583	0.637	0.809	0.629	1.385	0.547	0.552	0.619	0.745	0.598	**1.146**
	Detected Spam	-	-	-	-	-	-	1	1	2	2	3	-
	Detected Biased	-	-	-	-	-	-	3	0	1	1	1	-

estimated true label for every item as a whole, could be obtained. With spam removing and bias correction, the accuracy and F1 measure can be improved by 28.0%, and 27.0%, respectively, on image quality rating task. For project grading task, we could reduce the standard deviation and RMS error up to 18.6%, and 17.3%, respectively. The number of correctly detected spam workers and biased sloppy workers is also presented in the table. The multiple views approach only applies optimization based truth discovery framework without worker filtering for each view, thus no spam or biased workers recognized.

5.5 Chapter summary

In this chapter, we focus on estimating true labels for indecomposable complex scoring tasks. This type of task requires the crowd workers possess specialized domain skills, which usually is not the case on popular micro-task crowdsourcing platforms. Compared to decomposable complex tasks, these tasks cannot be decomposed in a straightforward manner. We propose a methodology to investigate workers' reliabilities on different views, which are labeling principles defined by experts to help crowds have a better grasp of the goals of a task. The main contributions of the research in this chapter are as next:

(1) A multiple views approach are proposed to assign varied weights on different views for each crowd worker. Most existing research infers truths by minimizing the overall errors on all the labeling criteria, which leads to a single weighted model – worker reliabilities are assumed to be identical on all the defined views. However, we give hypothesis that the workers' work performance might differ from one view to another. Based on the assumption, we develop the model to weight worker's labels on each view separately, in order to obtain high quality estimated

true labels for items.

(2) An iterative worker filtering approach is integrated into the proposed multiple views approach, to give further improvement for the quality of consensus view labels. The worker filtering process is able to remove the spam workers and correct labels given by biased workers, and it is applicable for both ordinal and numerical data types. It is built upon the optimization based truth discovery framework, which is utilized to estimate the true labels on each view. In other words, the integrated multiple views approach will tackle the spamming and biasing problems while inferring truths on each view within the defined views set. The consensus labels for the views on each item are then combined to achieve a single label for this item (overall label).

(3) Real world data sets on both ordinal and numerical label scale are collected to demonstrate the effectiveness of the proposed integrated multiple views model. Image labeling task request ordinal labels, on the scale of 1-5, from MTurk platform to reflect the quality of each image, while project grading task asks the workers/graders to evaluate students' project reports from a school class on the numerical scale. For each of the tasks, 5 different views are defined by experts, and provided to workers as labeling principles while accomplishing the tasks.

The analysis on the collected data sets showed that the proposed multiple views approach has a positive effect on the quality of results. The output of image labeling data set presented significant improvement on both of the calculated evaluation metrics. We could also reduce the RMS error on project grading data set by 10.6%. By integrating the worker filtering algorithm into the multiple views approach, further improvement is able to be obtained for the quality of the inferred labels for items. For example, the accuracy of the image labeling data set can be improved up to 28.0%, and the RMS error for project grading data set can be reduced up to 17.3% (on overall item labels instead of view labels).

The workers are investigated and analyzed in a single labeling task in this work. However, it is possible to memorize each worker's reliabilities on various views, and utilize this useful information for future work. When a labeling task comes in, only the top n workers on each view v_i are selected to give labels on it. Since less work has been done by the worker (gives label on one view instead of all views), less cost can be spent to complete the task, while maintaining the quality of the obtained labels. In the meanwhile, due to the fact that only reliable workers are chosen, fewer workers are then required to provide labels in order to achieve high quality estimated true labels, leading to less budget requested to complete the task.

CHAPTER 6

SEMI-SUPERVISED LEARNING FOR WORKER FILTERING IN CROWDSOURCING SYSTEMS

The frameworks proposed in previous chapters, such as SRDF, and ITSC-TD models, are under the hypothesis that no gold truth is available while inferring the true labels from observed data. Thus, the worker reliability estimation and truth discovery processes are done under unsupervised learning scenarios. In this chapter, we assume that a limited amount of gold data is able to be obtained, and we can benefit from applying this subset of data in the process of truth inference. The proposed framework adopts a semi-supervised learning approach to investigate the worker's performance in crowdsourcing applications. With better estimation of worker reliability, it is possible to obtain high quality consensus results from the crowdsourced noisy data. The experimental results demonstrate that substantial improvements can be achieved with the semi-supervised learning method in comparison to the unsupervised models.

6.1 Introduction

Crowdsourcing has provided a large pool of workers for solving large scale tasks, which could be easily completed by human intelligence, such as image labeling, relevance judging, or natural language processing, etc. The emerge of the convenient and efficient crowdsourcing platforms, such as Amazon Mechanical Turk (MTurk)[2] and CrowdFlower[3], has attracted more and more workers with varied backgrounds and intentions. One of the big challenges that exists in crowdsourcing systems is that it is difficult to control the quality of the crowds [50]. There is quite a lot of research has reported the existence of malicious workers, spam workers, and biased workers [31, 50, 60, 75] among the crowds. As indicated in [44], "accuracy of individual crowd workers has often exhibited high variance in past studies due to factors like poor design or incentives of tasks, ineffective or unengaged workers, or task complexity". In addition, the crowd workers usually have diverse education levels, preferences, and motivations, leading to unknown work performance while accomplishing the labeling

[2]https://www.mturk.com/
[3]https://www.figure-eight.com/data-for-everyone/

tasks. Consider for example, a worker is only interested at the monetary reward for completing a task, so he might give low quality feedback with little effort and time.

To ensure the quality of the answers submitted by crowd workers, several approaches have been proposed to combat the challenge. 1) One common method for quality control is to request labels from multiple workers for one item, which is called repeated labeling. The labels are then aggregated to reach a single label,which is the consensus result, for this item. One simple aggregating method is Majority Voting (MV) [41, 43], in which the answer that receives the maximum number of votes is treated as the final label. However, in MV, all workers are assumed to have the same reliability, which is not the case in most scenarios. Another popular approach for aggregating the multiples labels for an item is denoted as weighted voting based on worker performance. In weighted voting, each worker's reliability will be evaluated, and the answers given by workers with high reliabilities will have big chance to be chosen as the true labels. 2) Worker filtering is also an efficient way for quality control in crowdsourcing systems. The workers with poor quality feedback are identified and excluded for the process of estimating the final results. For example, a worker which assigns random answers for the items are eliminated as spam worker from the workers group, since there is no contribution dedicated to the inference of the true labels. In general, worker filtering models aim to distinguish between reliable and noisy workers among the crowd.

The worker filtering systems are built upon the investigations of the workers' performance. There are mainly 3 different ways to conduct the performance analysis: a) Pre-task work quality inspection. The workers are requested to accomplish a set of pre-defined questions before proceeding to the real labeling tasks. The answers, which are the gold truths, to the prerequisite questions set are already known, so that workers' feedback is able to be evaluated. Only the workers who fulfill the specified performance requirements are permitted to work on the actual tasks. For example, MTurk provides a pre-qualification system to assess the skill level of a prospective worker. b) Post-task work performance analysis. The quality of the answers provided by a worker is not assessed until a task has been accomplished. Different from the pre-task approach, any worker who is interested in working on the task will be allowed in post-task analysis systems. There have been proposed worker filtering methodologies [31, 40, 50, 60] built upon the post-task analysis, which detect and exclude the low quality workers to consolidate the labels for items. The models are often carried out by the unsupervised learning approach, in which with the hypothesis that no gold truths are known. Take the research in [60] as an example, the authors separate the low quality workers (spam workers) from reliable workers, by comparing the estimated true labels with the feedback obtained from the

TABLE 6.1

Worker performance analysis methods for worker filtering systems

	Description
Pre-task	Workers are required to accomplish a set of pre-defined questions, with answers know, before working on the actual task. Worker's performance is assessed on the pre-qualification questions set.
Post-task	Worker's performance is evaluated after collecting all the labels from the crowd for a labeling task. The models are able to be carried out in the unsupervised settings (without gold truths known).
Time-series	Time-series model takes into account the temporal component while estimating worker's performance. Predictive models are developed to represent the time-varying accuracy of the workers.

labelers. The spam workers are then removed from the crowd along with their labels. c) Time-series worker model. Time series model takes into account the temporal correlation in task performance of the workers. Predictive models are developed to estimate the crowd worker's performance, based on various features including the time component. Consider for example, while predicting the worker accuracy at time t, the value of accuracy at time $t-1$ is utilized together with other features such as the time spent on labeling the item, and the worker's familiarity of the task topic, etc [162]. Another example could be a Sequential Bayesian Estimation framework proposed by Donmez et al. [163], to estimate the expected worker accuracy at every time step. The framework relies on the Hidden Markov Model (HMM) to represent the time-varying accuracy of the workers, and applies particle filtering method to infer the expected accuracy. Table 6.1 presents the summarization of the discussed worker analysis models.

The work in this chapter focuses on the post-task worker performance analysis, and filters the low quality workers based on the analyzed results. A variety of existing approaches have been proposed to consider worker reliabilities and accuracies without using any gold truths [31, 50, 60]. However, if true labels of some items are provided, worker reliability estimation can be effectively informed and guided, by utilizing these truths. Better consensus results can be achieved when the workers' reliabilities or performance are better studied, in weighted voting models. Snow et al. [42] proposed a fully-supervised Naive Bayes (NB) approach to estimate the true labels and conduct bias correction for non-expert workers, in their work. In a fully-supervised model, the presence of full set of gold truths are required to conduct the truth discovery process. This is unrealistic in most cases, especially for large scale labeling tasks (experts labels are too expensive to get). A good way to get high quality consensus labels is to introduce a small set of gold truths. As such, the worker reliability estimation and true label inference procedure could benefit from the limited supervision.

This chapter presents a novel way of obtaining high-quality consensus results with limited supervision. The assumption is made that a set of limited gold labels, which are the answers offered

by experts, are provided as the available information. The proposed framework first eliminates the low quality workers with a semi-supervised learning model, by utilizing the small set of gold truths. The consolidated labels from reliable workers are then used for inferring the true labels for items. The idea is motivated by the feature selection while building a predictive or classification model, in which only the features that are most useful or most relevant will be chosen. The labels provided by workers with poor performance are useless and can decrease the accuracy of the final consensus results, thus it is helpful to filter them. As a result, only the answers from reliable workers, which are the most useful labels, are selected. The low quality workers are defined as the workers who offer random labels for items, and they are referred as spam workers. To infer the true labels after filtering the spam workers, an optimization based truth discovery framework [132] is applied. To investigate the proposed approach, both synthetic and real world data is used to test the quality of the consensus results.

The rest of the chapter is organized as follows: Section 6.2 presents the related work in the domain of quality control for crowdsourcing systems. Section 6.3 presents the proposed framework which integrates the semi-supervised worker filtering model into the truth discovery method. Experimental results and analysis are presented in Section 6.4. Section 6.5 presents the summary of this chapter.

6.2 Literature review on quality control for crowdsourcing applications

The nature of crowdsourcing tasks, which are tedious, anonymous, and typically low rewarded, makes it necessary to have quality control of the data created. A lot of research has been done to improve the quality of the crowdsourced data [31, 41, 44, 50, 60, 63]. The work can be classified into different categories: use redundant workers (repeated labeling) and aggregation, worker filtering, and implement crowdsourcing systems with better task design.

6.2.1 Related work on redundant work & aggregation

Redundant work, which is also known as repeated labeling, asks multiple workers to accomplish the same crowdsourcing task. Sheng et al. [41] examined the impact of repeated labeling in data quality, and they considered the effect of repeated labeling on the accuracy of supervised modeling. Dawid and Skene [46] used multiple noisy labelers to estimate the individual error-rates, using the Expectation Maximization (EM) algorithm. Nowak and Rüger [164] explored the effect of repeated annotation on the quality of image annotation tasks. They conducted a study on inter-annotator

agreement for an image corpus with multiple labels. The work demonstrated that repeated labeling could make it possible to improve the data quality. The studies in [42, 165, 166] also found that the results quality can be promising while performing the repeated labeling on distributed data annotation.

One problem associated with redundant work is that large scale redundancy is expensive. To determine the number of labels for each item, various strategies have been proposed by prior research [41, 42, 47, 48]. The easiest approach is to pre-define the number of labels which is required for each item. Snow et al. [42] demonstrated in their work that an average of 4 non-expert labels per item is required to emulate the expert-level label quality. However, different crowdsourcing applications and the varied worker quality makes it impractical to define a specific number for redundant work. The consideration of data acquisition costs together with the data quality has seen increasing research attention [41, 48, 167]. Active learning [168] is one hot topic, which focuses on the problem of cost for label acquisition. Sheng et al. [41] examined the relationship between labeler quality, number of labels, and the overall quality of labeling with multiple labelers. They proposed cost-effective labeling, which is an active learning model based on the label uncertainty and model uncertainty. The items with low label and model uncertainty are selected to acquire additional labels. Laws et al. [48] evaluated the utility of active learning in entity recognition and sentiment detection crowdsourcing tasks. The work showed that active learning and crowdsourcing are complementary methods, for lowering annotation cost. Ambati et al. [167] proposed a new paradigm, which is called Active Crowd Translation (ACT), to combine active learning and crowdsourcing to enable translation for low-resource language pairs. Zhao et al. [169] proposed crowdsourcing annotations using an active learning method to select instances, by combining uncertainty and inconsistency measures. In addition, the work in [106,170–172] presented different ways to integrate active learning into crowdsourcing applications, in order to improve the obtained label quality.

To get a single label (consensus label) for each item from the multiple workers' answers, a variety of aggregation algorithms are studied in a lot of work. The methodologies in literature for answer/label aggregation could be classified into 3 different categories: 1) unsupervised consensus labeling, 2) supervised consensus labeling, and 3) semi-supervised consensus labeling.

Unsupervised consensus labeling approaches are able to carry out aggregation procedures without using any gold truths. The most straightforward approach, which is known as Majority Voting (MV) [41, 43], is to choose the answer with the maximum number of votes, as the consensus result. In MV, worker reliabilities are not considered while aggregating the crowdsourced labels.

To obtain high quality consensus results, weighted voting has drawn great research interest [45, 46, 71, 119, 128]. In weighted voting, worker reliabilities are taken into account, while estimating the true labels from observed data from crowds. Dawid and Skene [46] proposed an approach models a confusion matrix for each worker, as well as the class prior. They utilized Expectation Maximization (EM) to estimate the true labels. Whitehill et al. [45] developed GLAD to simultaneously infer the expertise of each worker, the difficulty of each item, and the most probable label for each item. Raykar et al. [71] used a Bayesian approach to add worker specific priors for each class. Their algorithm evaluates the different experts and gives an estimate of the actual hidden labels by using an automatic classifier. Zhou et al. [131] utilized a minimax entropy principle to estimate the true labels from the crowd answers. Their method assumes that labels are generated by a probability distribution over workers, items, and labels. In addition to the models built with probabilistic techniques, there is another type of algorithm could be utilized to obtain truths from crowd answers – optimization based truth discovery methods [119, 132]. Meng et al. [128] proposed an effective optimization based truth discovery framework to infer the truths for crowd sensing of correlated entities. Aydin et al. [119] investigated a novel weighted aggregation method to improve the accuracy of crowdsourced answers for multiple-choice questions. They deploy the optimization based truth discovery algorithm, as well as the light weight machine learning (ML) techniques for building more accurate crowdsourced question answering system.

Supervised consensus labeling is discussed by the work of Snow et al. [42]. They proposed a fully-supervised Naive Bayes (NB) approach to estimated the true labels, and conduct bias corrections for non-expert workers. Few discussions have been conducted for fully-supervised methods, due to the fact that it is unrealistic to have the full set of gold truths available for the items. Ipeirotis [173] compared the effectiveness of unsupervised and supervised methods for obtaining consensus labels.

Semi-supervised consensus labeling approaches estimate consensus results using a limited amount of expert labels. Tang and Lease [44] proposed a semi-supervised Naive Bayes (SNB) to infer the consensus labels, using both labeled and unlabeled items.

6.2.2 Related work on worker filtering

Although redundant work helps relieve the quality control problem with some extent, it is not panacea, which sometimes may not lead to good result [174]. Another way to help improve the quality of data obtained through crowdsourcing, is to study the worker performance or reputation

[31,50,52,53,77,163]. By excluding the low quality workers, it is possible to improve the performance of obtained consensus results. In Amazon Mechanical Turk, requesters are able to specify the workers' qualifications and approval rate. The low reputation workers are then forbidden by the system. Bernstein et al. [175] introduced a system which filters workers based on a set of gold standard questions, in which the answers are known. The worker's answer for the ground truth question must include at least one choice from the inclusion list and none from the exclusion list. Le et al. [77] used a set of training data, which is the gold standard data, to ensure the quality of worker judgments for search relevance evaluation. The workers are required to complete labeling a specified number of judgments on the training data. They block the workers with poor accuracy in the research. Hoßfeld et al. [176] detected unreliable workers using gold standard questions with care, while modeling Quality of Experience (QoE) for YouTube. These methodologies carry out pre-screening procedures to ban the workers who do not pass a test with gold standard data. Downs et al. [52] used two previously pilot tested screening questions, to disqualify MTurk workers who participate but don't take the study tasks seriously.

Some research conducts worker performance analysis after collecting the data from crowds. The models are built upon investigating of the labels obtained from the workers. Ipeirotis et al. [31] blocked the low-performing workers and spammers in their work. They presented a novel algorithm which separate the true (unrecoverable) error rate from the (recoverable) biases that some workers exhibit, via generating a scalar score representing the inherent quality of each worker. Raykar and Yu [50] proposed an empirical Bayesian algorithm, which is called SpEM, to eliminate spammers by defining a spammer score to rank the annotators. The consensus labels are obtained based only on the good annotators. Vuurens et al. [60] classified the workers into various categories based on their behavior characteristics, and rejected the workers who belong to the spammers group. The study examined the effect that spam has on the relevance judgments through crowdsourcing. The work in [10, 78] categorized the patterns of the workers' behavior, and then revealed the spam workers according to the patterns. Some aggregation approaches such as EM model proposed by Dawid and Skene [46], are able to automatically recognize the malicious workers in a binary labeling task. Paiement et al. [177] proposed to use inter-annotator agreement as a quality measure for MTurk labels. Only reliable workers are selected for the final results. Liu et al. [178] developed a novel robust personal classifier (RPC) to automatically learn an expertise score for each worker. The learned expertise score is then used to eliminate spammers or low-quality workers.

There is a set of solutions [49,53] for quality control by filtering the noise from collected labels,

instead of directly eliminating low-quality workers. Zhang et al. [49] proposed an adaptive voting noise correction algorithm (AVNC) to identify and correct the noises, with the help of estimated qualities of labelers provided by the ground truth inference. Chen et al. [53] utilized a probabilistic model to remove problematic data, by checking individual consistency and overall consistency of workers.

Time-series models [162, 163, 179] have also been proposed to investigate workers' performance. Donmez et al. [163] developed a framework based on sequential Bayesian estimation to learn the expected accuracy for workers at each time step. A variant of the particle filtering method is adopted the expected accuracy at every time step. Jung et al. [179] proposed a time-series label prediction model, which predicts the quality of each worker's next label, by summarizing past worker behavior. Jung and Lease [162] built a discriminative, generalizable feature-based model to estimate crowd assessor's next judgment correctness, in their work.

6.2.3 Related work on crowdsourcing task design

There has been some research demonstrated that high quality crowdsourced feedback could be obtained with better task design [63, 180]. Liu et al. [63] implemented a Crowdsourcing Data Analytics System (CDAS) to allow task design and deployment of various crowdosurcing applications. It deploys a quality-sensitive answering model, which provides an estimated accuracy for each generated results based on the human workers' historical performances. Hirth et al. [180] proposed two methods to detect cheating workers for crowdsourcing systems: a majority decision (MD) and a control group (CG) approach to re-check the main task. MD is used to obtain the result with the most workers voted. For CG, a worker works on a main task and a control group consisting of some other workers re-checks the result. Huang et al. [181] introduced a general approach for automatically designing tasks on Amazon MTurk. The constructed models are trained on worker feedback over a set of designs, and are then utilized to optimize a task's design. Bernstein et al. [54] proposed a Find-Fix-Verify crowd programming pattern, which splits tasks into a series of generation and review stages, to improve the worker quality.

Careful task design is particularly important for crowdsourcing complex tasks, which requires the workers possess domain specific skills, and more time and effort compared to micro-tasks. For example, the authors in [6, 12, 13, 19, 84, 148, 149] crowdsourced the complex tasks by breaking them into smaller subtasks that are easier to solve. Cheng et al. [148] explored the costs and benefits of decomposing macrotasks into microtasks for three task categories: receipt arithmetic, line sorting,

and audio transcription. It is found that breaking these tasks into microtasks results in longer overall task completion times, but higher quality outcomes and a better experience that may be resilient to interruptions. Kittur et al. [12] developed the CrowdForge framework and toolkit in their work for crowdsourcing decomposable complex tasks. Their approach builds on the general approach to simplified distributed computing exemplified by systems such as MapReduce [87] of breaking down a complex problem into a sequence of simpler subtasks, using a small set of subtask types and managing the dependencies between them. The benefits and limitations of CrowdForge is demonstrated on case studies of article writing, decision making, and science journalism. The authors in [149] focused on crowdsourcing itinerary plans as a case study and introduced a collaborative planning system called Mobi, which takes as input a planning mission containing a set of constraints articulated by the user, and produces as output an itinerary that satisfies the mission. In this planning system, the idea of breaking down a task into subtasks to request solutions is implicitly utilized for better results.

For the complex tasks which are indecomposable, defining expert rubrics while designing the tasks could be of great help in improving results' quality. Dow et al. [14] designed the Shepherd system to manage workflows for tasks posted on MTurk. Their work showed that crowds can be shepherd via offering expert rubrics, and task-specific rubric yields better results. Yuan et al. [4] evaluated the use of expert rubrics in helping crowd workers provide high quality feedback, with crowdsourcing the task of assessing weather UI dashboard designs. The work focused on the salient differences between experts and novice workers feedback. Saddler and good [97] suggested in their study that with the helping of rubrics (provided to students), there is a high correlation between students and their teacher on test questions. The work in [90–92] applied rubrics to help crowd workers to provide rapid and relevant feedback, in the area of design critique. Kulkarni et al. [161] found in their research that well-designed rubrics can help lower the variance and improve accuracy in the grading process, with peer and self assessment, for massive online classes. Luther et al. [91] presented the CrowdCrit crowdsourcing system, in which workers use rubrics of design principles to give critique statements for UIs created in design contests.

The work in this chapter concentrates on filtering workers with poor performance, and under the hypothesis that limited amount of gold truths are available. The method is inspired by the semi-supervised Naive Bayes (SNB) consensus labeling model introduced by Tang and Lease [44]. In [44], the proposed SNB model targeted towards obtaining "soft" labels for each unlabeled item, by using labeled ones. Different from the SNB, the proposed semi-supervised model in this chapter aims

at filtering low-quality workers, instead of inferring true answers for unlabeled items. In other words, SNB estimates true labels via semi-supervised learning algorithm, whereas our proposed model eliminates workers with low performance under the semi-supervised setting. Both of these two algorithms are developed for the purpose of improving the quality of consensus results.

6.3 Proposed semi-supervised worker filtering model & truth discovery for crowdsourcing systems

The proposed worker filtering model is developed as a semi-supervised learning algorithm for blocking the low-quality workers. Only the labels provided by reliable workers are utilized for inferring truths (truth discovery) for the unlabeled items. The details of the semi-supervised model for worker filtering is presented in Section 6.3.1. The overview of the framework for truth discovery combined with worker filtering approach is presented in Section 6.3.2.

6.3.1 The semi-supervised worker filtering method

In the context of this chapter, we are interested in the low quality workers, which is called spam workers here. The spam worker is defined as the worker with poor performance who assigns random labels to the items. We present a novel way to eliminate the spam workers by utilizing a small set of gold truths, in order to improve the consensus results. The labeling task discussed is assumed to have ordinal labels (labeling scale > 2 choices), which is referred as crowd scoring task. In particular, the items with gold truths known are called labeled items, and those ones without gold labels provided are denoted as unlabeled items.

Suppose that a crowd scoring task has a possible label set C consists of K ordinal labels $C = \{1, 2, 3, ..., K\}$, and there are total M items $E = \{e_m\}_{m=1}^{M}$. The gold label corresponding to each item is denoted as g_m. We assume that there are N workers $W = \{w_j\}_{j=1}^{N}$ recruited to complete the labeling task. Let the label assigned by worker j to item m be $y_m^j \in C$. Let n_{mk}^j denote the number of times item e_m receives response k from worker j. Assume $\{T_{mc}\}$ is a set of indicators for class (label) membership of item e_m, such that $T_{mt} = 1$ if t is the true label of e_m and $T_{mt} = 0$ otherwise. To filter the spam workers among the crowd, we adopt the spammer score approach proposed by Raykar and Yu [50] to characterize their behaviors. This is because the spammer score can be combined with the Expectation Maximization (EM) algorithm, to better utilize the information both offered by labeled and unlabeled items.

According to the definition of the spam workers, it could be known that they assign labels

randomly to items. Thus, the label given by spam worker is independent from the gold label:

$$Pr(y_m^j = k|g_m = c) = Pr(y_m^j = k|g_m = c') \tag{6.1}$$

Let $\alpha_{ck}^j = Pr(y_m^j = k|g_m = c)$, and $\alpha_{c'k}^j = Pr(y_m^j = k|g_m = c')$. According to Eq(6.1), $\alpha_{ck}^j = \alpha_{c'k}^j$ for a spam worker. Assume A^j is a $K \times K$ confusion matrix for worker j, with entries $[A^j]_{ck} = \alpha_{ck}$. When worker j is a spammer, the rows of A^j should be equal to one another. Consider for example, if the labeling task has label set with 3 ordinal labels, the confusion matrix for a spam worker could be $A^j = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.4 & 0.3 & 0.3 \\ 0.4 & 0.3 & 0.3 \end{bmatrix}$. A^j actually is a rank one matrix of the form $A^j = ev_j^T$, where $v_j \in \Re^K$ is column vector which satisfies $v_j^T e = 1$, and e is column vector of ones. In the example above, $v_j^T = [0.4, 0.3, 0.3]$. The distance (Frobenius norm) [50] of the confusion matrix to the closest rank one approximation can be presented as:

$$S^j = \|A^j - e\hat{v}_j^T\|_F^2 \tag{6.2}$$

where $\hat{v}_j$ solves:

$$\hat{v}_j = arg\min_{v_j} \left\|A^j - ev_j^T\right\|_F^2 \quad subject\ to \quad v_j^T e = 1 \tag{6.3}$$

Solving Eq(6.3) yields $\hat{v}_j = (1/K)A^{jT}e$, then from Eq(6.2), we could obtain:

$$S^j = \frac{1}{K}\sum_{c<c'}\sum_k (\alpha_{ck}^j - \alpha_{c'k}^j)^2 \tag{6.4}$$

For a worker with S^j close to 0, then he is a spammer. A perfect worker would have $S^j = K - 1$. By normalizing the score of S^j to lie between 0 amd 1, we have:

$$S^j = \frac{1}{K(K-1)}\sum_{c<c'}\sum_k (\alpha_{ck}^j - \alpha_{c'k}^j)^2 \tag{6.5}$$

The original spammer score approach is proposed to conduct spam worker elimination under unsupervised setting, in which no gold truth is utilized during the process. In order to benefit from the provided limited amount of gold labels, we investigate the EM [46] algorithm. The EM algorithm is proposed to estimate the error rates of each worker j using a latent confusion matrix $[\alpha_{ck}^j]_{K\times K}$, where the element α_{ck}^j denotes the probability of worker j gives label k to an item with the true label as c. It can be estimated from:

Algorithm 6.1 Semi-supervised worker filtering algorithm.

Input: A set of labels $\{y_m^j\}$ from worker j to item e_m, and a set of true labels $\{g_l\}$ to items $e_l \in L$. Threshold ε for filtering workers.

Output: Confusion matrix α_{ck}^j for each worker j, class prior distribution $\{p_k\}_{k=1}^K$, and the estimated spammer score S^j for each worker j.

1: Initialization: initialize the unlabeled items with majority voting with the labels provided by workers.
2: **repeat**
3: {**Semi-supervised worker filtering**}:
4: Given the true labels for items in L and estimated labels for items in U, estimate the latent model
5: parameters α_{ck}^j using Equation (6.6).
6: Estimate spammer score for each worker using Equation (6.5).
7: **for** $j = 1...N$ **do**
8: **if** Worker j's spammer score $< \varepsilon$, and without diagonal attribute. **then**
9: Eliminate worker j as spammer.
10: **end if**
11: **end for**
12: {**Optimization based truth discovery**}:
13: {Calculate the worker's weight}
14: Update worker's weight W using equation (4.24) to infer worker reliability, based on the estimated
15: truths.
16: {Calculate the estimated truths}
17: **for** $m = 1...M$ **do**
18: Update the truth of m^{th} item y_m based on observations and current weight estimations, from the
19: worker who contributed to this item, according to equation (4.25).
20: **end for**
21: **until** Convergence
22: Return the estimated true labels for items in set U.

$$\hat{\alpha}_{ck}^j = \frac{\sum_{m=1}^{M} T_{mc} n_{mk}^j}{\sum_{k=1}^{K} \sum_{m=1}^{M} T_{mc} n_{mk}^j} \tag{6.6}$$

and the class/label prior $\{p_k\}_{k=1}^K$ is estimated as:

$$\hat{p}_k = \sum_{m=1}^{M} {}^{T_{mk}}/_{M} \tag{6.7}$$

Based on the obtained Eq(6.6) and Eq(6.5), we could see that the output of the EM algorithm, which is the worker confusion matrix, can be utilized as input for spammer score method to filter the spam workers. To detect the spam workers, a threshold ε could be set for the estimated spammer score. If a worker's spam score $< \varepsilon$, then the worker is considered as low-quality worker, and could be removed. Otherwise the worker is treated as a reliable one.

The availability of a limited set of gold truths allows us to better estimate the confusion matrix for each worker, together with the unlabeled items. Assume the set of items with gold truths known is denoted as L, and the set of unlabeled items is referred as U. We propose the semi-supervised worker filtering method, which is presented in Algorithm 6.1 line 3-11. The true labels for items in L are utilized to estimate the workers' confusion matrix. The obtained confusion matrix is then applied to get a spammer score for each worker with Eq(6.5). The spam workers,

who give random labels for the items, are removed by comparing the calculated spammer score to the threshold ε.

6.3.2 The truth discovery framework with integrated semi-supervised worker filtering approach

In order to infer the true labels from the reliable workers' feedback, after eliminating spam workers, the optimization based truth discovery framework is applied. This framework is selected since it is easier to extend to truth discovery problems for numerical labeling tasks for future work. It is an iterative approach to infer truths, as well as the worker reliabilities, by minimizing the overall weighted deviation between the identified truths and the observed data (labels offered by workers). The details have been discussed in Chapter 4, Section 4.3.3. Eq(4.22) presents the formulation which needs to be optimized. The loss function, which is denoted as $d(\cdot)$ in Eq(4.22), is set with 0-1 loss.

The semi-supervised worker filtering algorithm is integrated into the truth discovery framework, to improve the consensus results for the items. The integrated framework is presented in Figure 6.1. Each worker is associated with one spammer score, which is estimated through the model parameters calculated for EM approach proposed by Dawid and Skene [46]. Spammer scores are compared with a predefined threshold to detect and eliminate spam workers, so that only the labels provided by reliable workers are utilized to infer the truths. To obtain the estimated true labels, the optimization based truth discovery framework [132], which consists of truth computation and worker weights update steps, is applied. The corresponding pseudo-code is shown in Algorithm 6.1. In the presented algorithm, line 3-11 corresponds to the semi-supervised worker filtering approach in Figure 6.1. The optimization based truth discovery method is described from line 12 to 20. The output of the methodology is a set of estimated truths for the unlabeled items ($\in U$).

As discussed in Section 6.3.1, theoretically speaking, the spammer score close to 0 indicates a spam worker, and a perfect worker tends to have the score close to 1. Defining threshold ε is critical for the worker filtering model. As stated in [50], it is difficult to decide what is the correct ε to use. Although precisely defining the threshold is an arduous task, it is possible to apply the discovery of Dawid and Skene [46] in their work, to assist the worker filtering procedure. According to the research in [46], in most cases, even though reliable workers have errors, their error will be on the diagonal of the true labels. It is referred as "diagonal attribute" of the reliable workers here. Table 4.2(a) gives an example of the discovery. In our work, we choose the threshold ε through binary search way, which is discussed in Section 4.3.2, for the data sets, and ε is set with the number

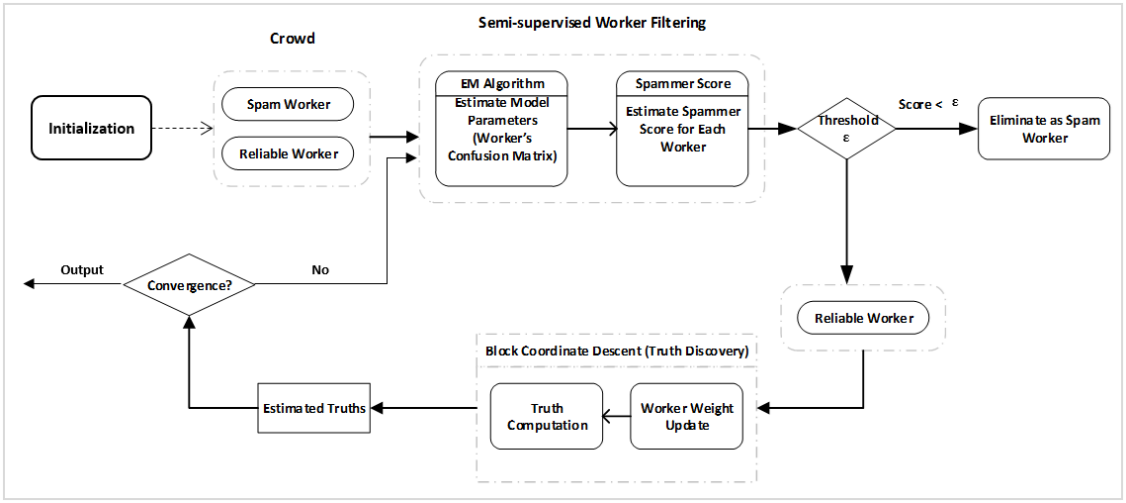


Figure 6.1: Overview of truth discovery framework integrated with semi-supervised worker filtering model

which gives the best results. A worker w_j is detected as a spam worker, if the estimated confusion matrix does not fulfill the diagonal attribute, and w_j's spammer score is less than ε. In other words, while a worker w_j presents the diagonal attribute, we always keep w_j for the truth discovery process, regardless of the estimated spammer score.

6.4 Experimental results and analysis

We present the experimental evaluation and analysis of the integrated Semi-supervised Worker Filtering – Truth Discovery (SWF-TD) framework, on both synthetic and real world data sets. Section 6.4.1 presents the experimental set up and the methods which are utilized for comparing the efficacy of the proposed methodology. Section 6.4.2 presents the experimental results and analysis on the simulated data set, and the evaluation of the output of the proposed semi-supervised worker filtering model, on real world data, is presented in Section 6.4.3.

6.4.1 Experimental setup

To demonstrate the impact of the semi-supervised worker filtering approach on the consensus results, experiments with the following methods are conducted:

(1) Unsupervised methods: Majority Voting (MV) which simply assigns the label with largest amount of votes to an item. When encountering a tie, we randomly select an answer from the voted results. Expectation Maximization (EM) model proposed by Dawid and Skene [46]

(EM-DS),which models a confusion matrix for each worker, and estimates the true labels for items. GLAD proposed in [45], which models the expertise of each worker, and the difficulty of each item in the task, to infer the true labels. Optimization based Truth Discovery (TD) [132], which applies a weighted voting scheme to obtain the consensus results. The results with worker filtering approach trained on unlabeled items only, integrated into optimization based truth discovery framework is also presented.

(2) Supervised methods: For supervised setting, the worker filtering approach is trained on the labeled items only. To aggregate the items collected from multiple workers, the optimization based truth discovery is applied.

(3) Semi-supervised methods: The proposed Semi-supervised Worker Filtering – Truth Discovery (SWF-TD) is utilized to give the experimental analysis.

The evaluation metrics adopted for measuring the performance of the experimental results are: accuracy and F1 measure. Accuracy gives the percentage of estimated labels which are the same as the gold truths, and F1 measure is a weighted average of precision and recall of an approach.

6.4.2 Experiments on synthetic data set

In this section, the experiments performed on synthetic data set are presented and analyzed, to show the benefit of utilizing the semi-supervised worker filtering algorithm. Section 6.4.2.1 gives the details about the generation of the simulated data. The experimental results and analysis is presented in Section 6.4.2.2.

6.4.2.1 Synthetic data set

The synthetic data set is generated as a crowd scoring task, with 5 different levels of labels in the possible label set $\{1, 2, 3, 4, 5\}$. A pool of 800 workers with differing accuracies are simulated, each with an accuracy generated according to the worker category. Varying ratios of spam workers are set for conducting the experiments. There are total 8000 items simulated and randomly assigned to each label class. The workers are randomly selected to give labels for the items. The number of workers for reviewing each item is randomly set between 2 to 8. The details of simulating reliable and spam workers, and their labels provided for items are presented as follows:

- Reliable worker: The accuracy for reliable worker is generated with a uniform distribution $U[0.6, 0.9]$. Consider for example, a worker w_k is assigned an accuracy parameter p_k. An item m, with true

TABLE 6.2

Accuracy and F measure calculated for varied proportions of spam workers under unsupervised setting

Unsupervised Models	Metrics	Proportion of Spam Workers				
		20%	40%	50%	60%	80%
MV	Accuracy	0.96	0.89	0.86	0.72	0.38
	F1	0.97	0.89	0.89	0.74	0.39
TD	Accuracy	0.97	0.94	0.92	0.75	0.41
	F1	0.97	0.95	0.93	0.76	0.45
EM-DS	Accuracy	0.98	0.94	0.93	0.77	0.43
	F1	0.97	0.95	0.94	0.79	0.45
GLAD	Accuracy	0.96	0.93	0.91	0.75	0.43
	F1	0.97	0.92	0.93	0.77	0.43
TD with Worker Filtering	Accuracy	**0.98**	**0.96**	**0.95**	**0.85**	**0.57**
	F1	**0.98**	**0.97**	**0.95**	**0.89**	**0.59**

label as g_m, is distributed to w_k to request a label. Based on p_k, we generate an label for the item: a) If a correct feedback should be given, then the gold truth is assigned to the item; b) Otherwise, a label is randomly selected from set $\{g_m - 1, g_m + 1\}$ for the item, according to the diagonal attribute of reliable workers. If $g_m - 1 < 1$, then choose $g_m + 1$. Similarly, if $g_m + 1 > 5$, then $g_m - 1$ should be selected as feedback.

- Spam worker: The accuracy for spam worker $\sim U[0.3, 0.6)$. Similar to reliable worker, the accuracy parameter determines whether a worker provides the correct label for an item. However, a different simulation process is performed for the spam worker, while a label different from gold truth should be provided, comparing to the reliable workers. In this scenario, we randomly select a label from the possible label set for the item as feedback.

6.4.2.2 Experimental analysis on synthetic data set

Experimental results for unsupervised worker filtering

All the evaluation metrics are reported as the average over running each experiment 10 times, in this section. We first investigate the performance of the unsupervised methods on the simulated data set. Varying percentages of spam workers are simulated to test the efficacy of the models. During the process, some items might have majority labels obtained from spam workers. As a result, there is not enough observed labels, which are the answers given by the workers, for these items, after applying the worker filtering algorithm. To combat this problem, we replenish the workers to make sure each item would have at least 2 observed labels. The results are shown in Table 6.2.

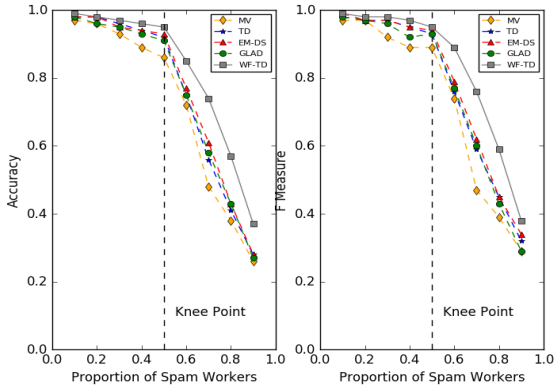


Figure 6.2: Accuracy and F measure for methodologies with varied spam worker ratios (unsupervised worker filtering)

(a) (b)

Figure 6.3: Improvement for accuracy and F1 measure comparing with and without worker filtering method (unsupervised)

In Table 6.2, the highlighted cells give the best results among all the methodologies. The MV approach shows the worst performance, and there is no significant difference between the results obtained from optimization based truth discovery framework, EM algorithm [46], and GLAD approach [45]. Based on the calculated evaluation metrics, it could be seen that better quality of consensus results are able to be obtained by eliminating the spam workers. For example, comparing the accuracy for TD and TD with worker filtering model, while 80% of the crowd are spam workers, 39.02% improvement can be finally achieved. To give a better illustration, Figure 6.2 presents the performance of the approaches on the synthetic data, through varying the ratios of spam workers from 0.1 to 0.9. The results in Figure 6.2 are consistent with Table 6.2: Although none of the gold truth is available, integrating worker filtering model into truth discovery procedure still shows a positive effect on the quality of consensus results. Figure 6.2 suggests that the proportion of spam

TABLE 6.3

Evaluation metrics estimated for TD and TD with supervised worker filtering

Models	Metrics	Proportion of Spam Workers				
		20%	40%	50%	60%	80%
TD	Accuracy	0.97	0.94	0.92	0.75	0.41
	F1	0.97	0.95	0.93	0.76	0.45
Supervised WF-TD	Accuracy	0.99	0.98	0.97	0.95	0.83
	F1	0.99	0.98	0.98	0.97	0.83

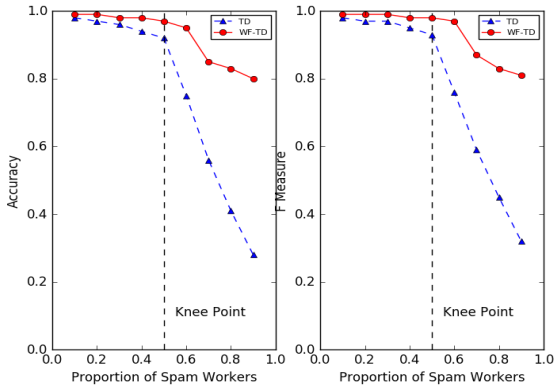


Figure 6.4: Accuracy and F measure for methodologies with varied spam worker ratios (supervised worker filtering)

workers equals to 0.5 is a knee point: While the proportion < 0.5, no significant improvement could be obtained after applying the worker filtering approach. In this scenario, most of the workers among the crowd are reliable workers, which leads to high quality observed labels for the items. If the proportion of spam workers > 0.5, the comparisons of the outcome between with and without worker filtering method present significant difference. Figure 6.3 indicates the percentage of improvement achieved for the calculated metrics, by comparing the TD combined with worker filtering to the other methodologies. As seen in Figure 6.3, only 3.3% and 2.2% improvement is observed for accuracy and F1 respectively, regarding to WF-TD (Worker Filtering - Truth Discovery) and TD, with 50% spam workers included in the crowd. However, increasing the ratio of spam workers to 80% in this case leads to up to 39.0% improvement on the accuracy, and 31.1% on the F1 measure.

Experimental results for supervised worker filtering

Next, we conduct experiments on the synthetic data with fully supervised worker filtering algorithm. All the generated items are set as labeled data and utilized to detect and eliminate the spam workers. For the purpose of inferring the true labels for items, optimization based truth

TABLE 6.4

Effects of different proportion of spam workers and varied rations of gold truths, on the consensus results

Methods	Ratio of Labeled Items	Proportion of Spam Workers							
		20%		50%		70%		80%	
		Accuracy	F1	Accuracy	F1	Accuracy	F1	Accuracy	F1
TD	-	0.97	0.97	0.92	0.93	0.56	0.59	0.41	0.45
SWF-TD	30%	0.97	0.97	0.95	0.95	0.78	0.79	0.59	0.61
	50%	0.98	0.98	0.95	0.96	0.80	0.82	0.65	0.66
	70%	0.98	0.99	0.96	0.97	0.83	0.84	0.74	0.77

discovery framework is performed. Although the worker filtering procedure, which estimates a spammer score for each worker, is conducted with the knowledge of gold truths, only the unsupervised truth discovery approach is used to infer the consensus labels. The output of MV is set as the estimated item labels in the initialization period. The results of supervised worker filtering approach, for different ratios of spam workers, is presented in Table 6.3.

The supervised worker filtering could recognize most of the spam workers, and as such is able to obtain high quality estimated truths from the noisy observed data. Even the ratio of spam workers is set as 0.8, the accuracy of the consensus results is maintained > 0.8. Similar to the outcome of unsupervised worker filtering, the calculated evaluation metrics show a significant difference between TD and supervised WF-TD , while the proportion of spam workers > 0.5. Table 6.2 and Table 6.3 intuitively showed that supervised worker filtering approach increases the accuracy of estimated labels in contrast to the unsupervised models. As an example, when spam workers proportion = 60%, the TD with unsupervised worker filtering achieved 13.3% higher accuracy than TD, whereas in supervised setting, WF-TD increased the accuracy by 26.7% compared to TD. Figure 6.4 gives the experimental results for supervised worker filtering method, by varying the ratio of spam worker from 0.1 to 0.9. It is indicated in Figure 6.4 that while spam worker proportion> 0.5, significant improvement can be obtained for both accuracy and F1 measure, after applying the supervised worker filtering model.

Experimental results for Semi-supervised Worker Filtering (SWF)

The influence of worker filtering approach with limited number of gold truths is investigated and evaluated here. Different proportions of spam workers and varied ratios of gold truths are set, to conduct the experiments. The items in the labeled data set (L) are randomly selected from the item set. For each ratio of the labeled items, the simulation process has an iteration of 20 runs to

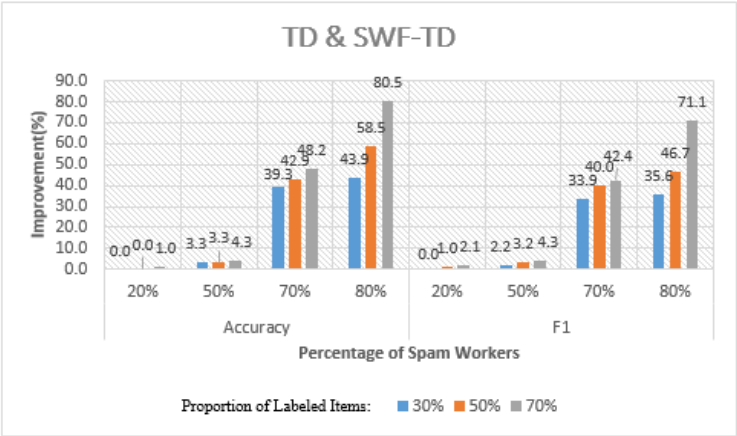


Figure 6.5: Improvement on evaluation metrics comparing TD and SWF-TD, with varied proportions of spam workers and gold truths

get more general and convincing results. Table 6.4 presents the effects of increasing the percentage of labeled items for worker filtering model from 30% to 50%, and 70%. It is demonstrated in Table 6.4 that while increasing the ratio of gold truths as prior knowledge, which is utilized to recognize the spam workers, better quality estimated labels could be obtained.

From Table 6.4, it can be seen that the accuracy and F1 measure are not significantly affected by the number of gold truths, when the proportion of spam workers ≤ 0.5. Figure 6.5 shows the improvement calculated (TD & SWF-TD) for the obtained evaluation metrics from Table 6.4. To better illustrate the impact of semi-supervised worker filtering approach on the consensus results, Figure 6.6 presents the output of unsupervised, supervised, and also semi-supervised models. In Figure 6.6, supervised worker filtering shows the best results, however, it requires great amount of expert labels, which is unrealistic in many cases. The availability of limited gold truths helps to improve the quality of estimated true labels. In particular, significant improvement could be obtained when there is a big proportion of spam workers, through applying the proposed worker filtering method. Compared to unsupervised scenario, semi-supervised approach presents much higher accuracy (spam worker proportion>0.5), with 50% to 70% gold labels known.

6.4.3 Experiments on real world data set

In this section, we evaluate the performance of the semi-supervised worker filtering method on two different real world data sets. One of the data sets, namely AdultContent2 (AC2), is publicly available, and it was originally used in [31]. The AC2 dataset includes AMT judgments for websites for the presence of adult content on the page, and its detailed description is discussed in Chapter

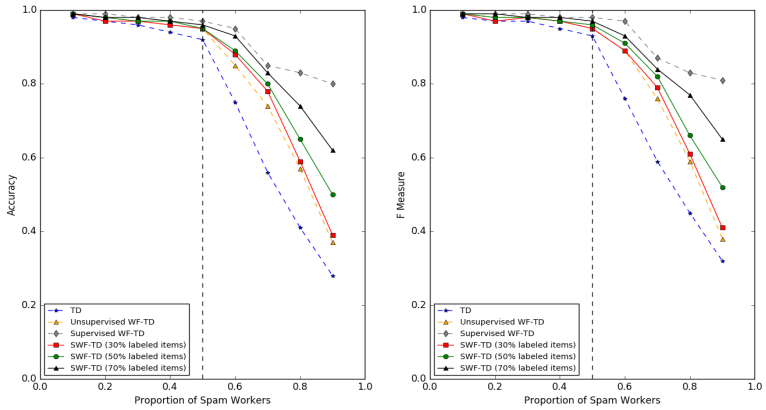


Figure 6.6: Experimental results for TD, unsupervised WF-TD, supervised WF-TD, and SWF-TD on the synthetic data set

4, Section 4.4.3. The second dataset is an image labeling dataset collected through MTurk. The original data entails a set of URLs for animal images, which is downloaded from CrowdFlower. We randomly select 500 out of the 3601 image URLs to collect labels from MTurk. The illustration of the dataset is presented in Chapter 5, Section 5.4.1.

Similar to the synthetic dataset, unsupervised and supervised methodologies are experimented on these two real world datasets first. As discussed in Section 6.4.2.2, it is possible to have items which do not possess enough labels from workers, in order to infer the truths after blocking spam workers. In contrast to the experiments for synthetic dataset, in which we replenish workers for such items, we simply discard these items (with less than 2 workers labeled). The results for all different truth inferring models, including integrated unsupervised worker filter – truth discovery framework, and supervised worker filtering – truth discovery approach, are presented in Table 6.5. The highlighted number represents the calculated metric for models combined with proposed worker filtering method. For both the AC2 and image labeling dataset, the results indicate that higher quality is able to be obtained after performing the worker filtering procedure (both unsupervised and supervised).

First of all, we give analysis for the unsupervised worker filtering approach. In Table 6.5, only 0.53% improvement is observed for accuracy AC2 dataset, by comparing unsupervised WF-TD to the TD or EM-DS methodology. It is due to the reason that the majority of the workers are reliable. As such, eliminating spam workers does not show significant impact on the final output. However, as opposed to AC2 dataset, the vast majority workers in image labeling dataset belong to spam worker group. Conducting worker filtering in the process of truth discovery results in up

TABLE 6.5

Accuracy and F measure for various truth inference models, unsupervised WF-TD, and supervised WF-TD (Worker Filtering-Truth Discovery)

Method	AC2		Image Labeling	
	Accuracy	F1	Accuracy	F1
MV	0.756	0.725	0.190	0.167
TD	0.759	0.735	0.214	0.206
EM-DS	0.759	0.736	0.238	0.226
GLAD	0.752	0.731	0.202	0.196
Unsupervised WF-TD	**0.763**	**0.740**	**0.315**	**0.302**
SWF-TD Labeled Ratio=0.4	**0.769**	**0.747**	**0.482**	**0.480**
Supervised WF-TD	**0.778**	**0.761**	**0.536**	**0.521**

to 33.6% improvement, when compared to EM-DS model, which shows the best results among the MV, TD, and GLAD models.

The supervised worker filtering – truth discovery framework outperforms the unsupervised WF-TD, on both of the real world datasets. For example, 2.0% and 70.2% improvement can be achieved, with supervised WF-TD, for the accuracy of AC2 and image labeling data respectively, when compared to unsupervised WF-TD. It is possible to combine the worker filtering process with other methodologies, such as EM-DS and GLAD model. However, the estimated evaluation metrics on TD, EM-DS, and GLAD indicate that there is no significant difference between the quality of the obtained consensus results, using these approaches. In addition, the optimization based truth discovery framework can be easily extended to infer the true labels for numerical data. As such, only the results for integrated WF-TD model are given here.

To evaluate the efficacy of the proposed semi-supervised worker filtering approach, we perform experiments on the AC2 and image labeling dataset, with varying amount/ratio of labeled items (from 0.1 to 0.9). The labeled items are randomly chosen from the item set. In addition, each ratio of the labeled items are experimented with the proposed methodology for 20 runs, and the final results are given as the average of these 20 runs. Figure 6.7 presents the accuracy and F1 measure calculated for the consensus results obtained, using the semi-supervised worker filtering model. While the proportion of labeled items= 0, it actually is the result for unsupervised WF-TD method. The output for supervised WF-TD framework is shown at the point with $x = 1.0$ in the figure. By analyzing the obtained metrics in Figure 6.7, it can be seen that the quality of estimated truths are further improved, when increasing the ratio of labeled items. Although the best results

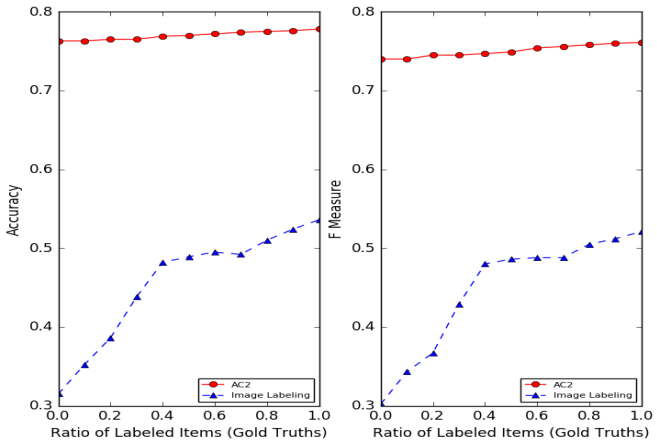


Figure 6.7: Accuracy and F measure for real world datasets, with varying ratios of labeled items

are achieved at $x = 1.0$ for both of the datasets, the semi-supervised WF-TD (SWF-TD) framework provide similar performance to the fully labeled model, when the proportion of available gold truths ≥ 0.4. Consider the ratio of labeled items (gold truths)$= 0.4$, the accuracy for SWF-TD is 0.769 for AC2 (0.778 with fully labeled model), and 0.482 (0.536 with fully labeled model).

To determine the threshold for filtering the spam workers in the proposed semi-supervised worker filtering approach, it is suggested to utilize the limited labeled items/gold truths, which are available for inferring the true labels. While selecting a threshold, an element from range [0.1, 0.5], which gives the best results regarding to the labeled items, can be chosen for the semi-supervised worker filtering procedure.

6.5 Chapter summary

In this chapter, the semi-supervised worker filtering (SWF) approach is proposed to integrate into the optimization based truths discovery framework. The framework is able to eliminate the spam workers, and lead to high quality consensus results for the items. The proposed SWF model utilizes a subset of items with gold truths known as a priori, to estimate a spammer score for each crowd worker. The spam workers are blocked by setting a threshold for the calculated spammer scores. Experiments on both synthetic and real world (MTurk) data are performed to demonstrate the efficacy of the proposed SWF approach. The results show that with limited supervision, significant benefit could be achieved in the process of detecting spam workers. Especially in the scenario that a large proportion of spam workers exist among the crowd, SWF presents a positive effect on the quality of estimated truths. Even when the amount of items with gold truths known is limited, it

is still possible to obtain similar consensus results of the fully supervised method, by applying the semi-supervised worker filtering approach. For example, the results on the AC2 and image labeling datasets presented similar accuracies to the supervised worker filtering method, via SWF approach, while the labeled item ratio is around 40%.

Removing the spam workers so that only reliable ones are allowed to complete the labeling task, leads to reduced cost. This is because fewer workers need to be recruited to obtain high quality consensus results, when most of the workers are reliable (demonstrated by the experimental results in Chapter 3, Section 3.6.2.2). The labeling tasks with ordinal scale have been discussed and experimented here. An interesting direction of the future work could be extending the semi-supervised worker filtering – truth discovery framework to numerical labeling tasks.

CHAPTER 7

CONCLUSIONS AND POTENTIAL FUTURE WORK

7.1 Conclusions

Guaranteeing the quality of data collected from crowds has always been a challenging topic for crowdsourcing systems, and it has been discussed by many of the studies and research. To obtain high quality crowdsourced data, it is desirable to have the majority of the crowds constituted of reliable workers to accomplish the tasks. The nature of crowdsourcing tasks, often tedious, decontextualized, and small rewarded, makes it difficult to attract high quality workers. In this dissertation, the focus has been put on investigating the workers' performances and removing those that are less desirable. Most of the existing work on worker filtering concentrates on binary labeling tasks, and does not distinguish between spam and biased workers. Spam workers randomly choose labels from possible answer set for items, whereas biased workers show specific error patterns, which could be utilized to obtain high quality estimated labels. We extend the tasks to ordinal and even numerical label scales, and propose to differentiate various worker categories, including spam and biased workers, in order to achieve high quality consensus results. The following frameworks are presented in the dissertation in the domain of worker filtering:

(1) Spam Removing and De-biasing Framework (SRDF)

The SRDF framework is introduced to distinguish between spam and biased worker groups, for the purpose of improving the quality of estimated true labels. Instead of binary labeling tasks, the work has been done for the crowdsourcing applications with numerical label scales in Chapter 3. Three different types of spam workers are proposed to be eliminated from the crowd, including random spammers, uniform spammers and sloppy spammers. It can be seen that significant improvement can be obtained for consensus results, by detecting and removing these spam workers. Another benefit of applying the spam filtering algorithm is that fewer workers, which lead to less budget, would be required to achieve high quality estimated truths. In order to tackle the biasing problem within the crowd workers group, an iterative bias detection method is developed to recognize the biased workers, and to correct their provided labels. This algorithm is built based on identifying

and reducing the consistent and similar behavioral errors, which reside within the labels provided by the biased workers. As such, a positive impact is presented in terms of the selected evaluation metrics (standard deviation, correlation coefficient, and RMSE). Integrating the spam filtering and bias detecting approaches into aggregating algorithms results in the proposed SRDF methodology. The conducted experiments demonstrate that SRDF outperforms all the baseline models.

(2) Iterative Self Correcting – Truth Discovery (ITSC-TD) Framework

The ITSC-TD framework is proposed to tackle sloppiness, which denotes the phenomena of observed labels fluctuating around the true labels. Sloppiness, in many cases, might degrade the quality of estimated truths. Different from labels provide by spam workers, which are independent from truths, the data obtained from sloppy workers (workers with sloppy behaviors) could still be utilized to infer accurate consensus results. It is, therefore, necessary to separate various types of workers, including sloppy and spam labelers. Chapter 4 presents a novel hierarchical classification of worker groups, and proposes to extract useful information from labels provided by a subgroup of sloppy workers – biased sloppy workers. The work focuses on the crowd scoring tasks, which are crowdsourcing tasks with ordinal labels. To detect the biased sloppy workers, a probabilistic based bias detection model is developed here. Finally, we integrate the bias detection model into an optimization based truth discovery framework (ITSC-TD), to derive the true labels. The integrated approach has demonstrated its superiority over a set of state-of-art methodologies, on both synthetic and real world data sets.

(3) Mutliple Views Approach with ITSC-TD Framework

The workers' reliabilities in different features, which are called views, of a complex task are investigated in Chapter 5. A multiple views approach is proposed to analyze worker's performance on each view, under the hypothesis that workers' performance might differ from one view to another. It assigns varied weights on different views for each of the workers. The ITSC-TD framework proposed in Chapter 4 is then integrated into the multiple views approach, to derive consensus view labels. To demonstrate the efficacy of the multiple views approach, real world data sets on both ordinal and numerical label scales are collected for experimental analysis. Image labeling data is obtained from the MTurk platform to reflect the quality of each image, on an ordinal label scale. While the project grading data set contains the assessments of students' project reports from a school class, on a numerical scale.

(4) Semi-supervised Worker Filtering – Truth Discovery Framework

A semi-supervised learning approach is discussed in Chapter 6, to investigate workers' performance in crowdsourcing systems. The assumption is made that a small set of gold truths is available as priori while inferring the true labels. The items with known gold truths are denoted as labeled items, otherwise are called unlabeled items. The labeled items are first utilized to estimate model parameters, and then to calculate a spammer score for each worker. Low quality workers are eliminated by comparing the spammer scores to a predefined threshold. The optimization based truth discovery framework is used to estimate the consensus results, from the labels provided by the remaining workers (after spam filtering procedure). It is demonstrated that similar results of the fully supervised method can be obtained by applying the semi-supervised worker filtering model.

7.2 Potential future work

Worker filtering algorithms, which eliminate the spam workers and detect the biased workers, are proposed and presented in both unsupervised and semi-supervised scenarios, in this dissertation. The purpose for such models is to improve the quality of crowdsourced labels obtained for the items. One potential direction of the future work is to extend the multiple views approach into a time-varying model. The developed multiple views model is utilized to better estimate the true labels, for indecomposable complex scoring tasks through crowdsourcing. Each view is defined as a rubric or criteria to guide the novice workers in completing the task. By including each worker's reliabilities on the expert views from prior work, it is possible to better model the worker performance over time. As such, high quality estimated truths are able to be obtained.

It is also interesting to integrate the bias detection model into the semi-supervised worker filtering (SWF) – truth discovery (TD) framework. The SWF-TD is proposed to eliminate the spam workers using a subset of labeled items (with gold truths known), while inferring the true labels. Extending the framework to deal with biased workers could be beneficial for further quality improvement of the consensus labels. In particular, it is of great importance to incorporate the bias detection approach, in the case that a large group of biased workers exists among the crowd.

Another direction of future work will utilize the iterative self correcting – truth discovery (ITSC-TD) framework, and the semi-supervised worker filtering – truth discovery (SWF-TD) methodology for mixed ordinal and continuous data. The efficiency of improving the consensus label quality, by using ITSC-TD and SWF-TD, is demonstrated with the crowdsourced data on ordinal scales. Deploying these two frameworks, for crowdsourcing applications with mixed data types, would be a potential extension in future research.

REFERENCES

[1] Thomas W Malone, Robert Laubacher, and Chrysanthos Dellarocas, "Harnessing crowds: Mapping the genome of collective intelligence," 2009.

[2] Eric Schenk and Claude Guittard, "Towards a characterization of crowdsourcing practices," *Journal of Innovation Economics & Management*, , no. 1, pp. 93–107, 2011.

[3] Wikipedia, "Expectation-maximization algorithm," 2016.

[4] Alvin Yuan, Kurt Luther, Markus Krause, Sophie Isabel Vennix, Steven P Dow, and Bjorn Hartmann, "Almost an expert: The effects of rubrics and expertise on perceived value of crowdsourced design critiques," in *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*. ACM, 2016, pp. 1005–1017.

[5] Chris Piech, Jonathan Huang, Zhenghao Chen, Chuong Do, Andrew Ng, and Daphne Koller, "Tuned models of peer assessment in moocs," *arXiv preprint arXiv:1307.2579*, 2013.

[6] Nathan Van Houdnos, "Can the internet grade math? crowdsourcing a complex scoring task and picking the optimal crowd size," 2011.

[7] Luca De Alfaro and Michael Shavlovsky, "Crowdgrader: A tool for crowdsourcing the evaluation of homework assignments," in *Proceedings of the 45th ACM technical symposium on Computer science education*. ACM, 2014, pp. 415–420.

[8] Jing Gao, Qi Li, Bo Zhao, Wei Fan, and Jiawei Han, "Truth discovery and crowdsourcing aggregation: A unified perspective," *Proceedings of the VLDB Endowment*, vol. 8, no. 12, pp. 2048–2049, 2015.

[9] Thierry Buecheler, Jan Henrik Sieg, Rudolf M Füchslin, and Rolf Pfeifer, "Crowdsourcing, open innovation and collective intelligence in the scientific method-a research agenda and operational framework.," in *ALIFE*, 2010, pp. 679–686.

[10] Dongqing Zhu and Ben Carterette, "An analysis of assessor behavior in crowdsourced preference judgments," in *SIGIR 2010 workshop on crowdsourcing for search evaluation*, 2010, pp. 17–20.

[11] Ece Kamar, Ashish Kapoor, and Eric Horvitz, "Identifying and accounting for task-dependent bias in crowdsourcing," in *Third AAAI Conference on Human Computation and Crowdsourcing*, 2015.

[12] Aniket Kittur, Boris Smus, Susheel Khamkar, and Robert E Kraut, "Crowdforge: Crowdsourcing complex work," in *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, 2011, pp. 43–52.

[13] Jon Noronha, Eric Hysen, Haoqi Zhang, and Krzysztof Z Gajos, "Platemate: crowdsourcing nutritional analysis from food photographs," in *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, 2011, pp. 1–12.

[14] Steven Dow, Anand Kulkarni, Scott Klemmer, and Björn Hartmann, "Shepherding the crowd yields better work," in *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*. ACM, 2012, pp. 1013–1022.

[15] Kevin D Ashley and Ilya M Goldin, "Toward ai-enhanced computer-supported peer review in legal education.," in *JURIX*, 2011, pp. 3–12.

[16] Fei Mi and Dit-Yan Yeung, "Probabilistic graphical models for boosting cardinal and ordinal peer grading in moocs.," in *AAAI*, 2015, pp. 454–460.

[17] Christopher J Brady, Andrea C Villanti, Jennifer L Pearson, Thomas R Kirchner, Omesh P Gupta, and Chirag P Shah, "Rapid grading of fundus photographs for diabetic retinopathy using crowdsourcing," *Journal of medical Internet research*, vol. 16, no. 10, 2014.

[18] 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*, vol. 17, no. 2, pp. 1–16, 2016.

[19] Aniket Kittur, Jeffrey V Nickerson, Michael Bernstein, Elizabeth Gerber, Aaron Shaw, John Zimmerman, Matt Lease, and John Horton, "The future of crowd work," in *Proceedings of the 2013 conference on Computer supported cooperative work*. ACM, 2013, pp. 1301–1318.

[20] Merriam-Webster Dictionary, "Entry: Crowdsourcing," *Online at http://www.merriam-webster.com/dictionary/crowdsourcing*, 2012.

[21] James Surowiecki, *The wisdom of crowds: Why the many are smarter than the few and how collective wisdom shapes business, economics, societies and nations*, Anchor, 2005.

[22] Henri Simula, "The rise and fall of crowdsourcing?," in *System sciences (HICSS), 2013 46th Hawaii international conference on*. IEEE, 2013, pp. 2783–2791.

[23] Mobenzi, "Human intelligence tasks," *Online at http://www.mobenzi.com/intelligence/index.php/types-of-tasks/*, 2011.

[24] Nathan Eagle, "txteagle: Mobile crowdsourcing," in *International Conference on Internationalization, Design and Global Development*. Springer, 2009, pp. 447–456.

[25] Michael R Merrifield and Donald G Saari, "Telescope time without tears: a distributed approach to peer review," *Astronomy & Geophysics*, vol. 50, no. 4, pp. 4–16, 2009.

[26] Aniket Kittur, Ed H Chi, and Bongwon Suh, "Crowdsourcing user studies with mechanical turk," in *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, 2008, pp. 453–456.

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

[28] Wikipedia, "Wikipedia," *Online at https://en.wikipedia.org/wiki/Wikipedia*, 2016.

[29] L Frederic, "For the love of mapping data," *TechCrunch*, 2014.

[30] Frederik Ramm, Jochen Topf, and Steve Chilton, *OpenStreetMap: using and enhancing the free map of the world*, UIT Cambridge Cambridge, 2011.

[31] Panagiotis G Ipeirotis, Foster Provost, and Jing Wang, "Quality management on amazon mechanical turk," in *Proceedings of the ACM SIGKDD workshop on human computation*. ACM, 2010, pp. 64–67.

[32] Jeff Howe, *Crowdsourcing: How the power of the crowd is driving the future of business*, Random House, 2008.

[33] 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*. Association for Computational Linguistics, 2011, pp. 1220–1229.

[34] Maria Antikainen, Marko Mäkipää, and Mikko Ahonen, "Motivating and supporting collaboration in open innovation," *European Journal of Innovation Management*, vol. 13, no. 1, pp. 100–119, 2010.

[35] Valérie Chanal and Marie-Laurence Caron-Fasan, "The difficulties involved in developing business models open to innovation communities: the case of a crowdsourcing platform," *M@ n@ gement*, vol. 13, no. 4, pp. 318–340, 2010.

[36] Anhai Doan, Raghu Ramakrishnan, and Alon Y Halevy, "Crowdsourcing systems on the world-wide web," *Communications of the ACM*, vol. 54, no. 4, pp. 86–96, 2011.

[37] Daren C Brabham, "Crowdsourcing the public participation process for planning projects," *Planning Theory*, vol. 8, no. 3, pp. 242–262, 2009.

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

[39] Jeffrey Heer and Michael Bostock, "Crowdsourcing graphical perception: using mechanical turk to assess visualization design," in *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, 2010, pp. 203–212.

[40] Hyun Joon Jung and Matthew Lease, "Improving consensus accuracy via z-score and weighted voting.," in *Human Computation*, 2011.

[41] Victor S Sheng, Foster Provost, and Panagiotis G Ipeirotis, "Get another label? improving data quality and data mining using multiple, noisy labelers," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 614–622.

[42] 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*. Association for Computational Linguistics, 2008, pp. 254–263.

[43] Hui Yang, Anton Mityagin, Krysta M Svore, and Sergey Markov, "Collecting high quality overlapping labels at low cost," in *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2010, pp. 459–466.

[44] Wei Tang and Matthew Lease, "Semi-supervised consensus labeling for crowdsourcing," in *SIGIR 2011 workshop on crowdsourcing for information retrieval (CIR)*, 2011, pp. 1–6.

[45] 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*, 2009, pp. 2035–2043.

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

[47] Yan Yan, Romer Rosales, Glenn Fung, and Jennifer G Dy, "Active learning from crowds.," in *ICML*, 2011, vol. 11, pp. 1161–1168.

[48] Florian Laws, Christian Scheible, and Hinrich Schütze, "Active learning with amazon mechanical turk," in *Proceedings of the conference on empirical methods in natural language processing*. Association for Computational Linguistics, 2011, pp. 1546–1556.

[49] Jing Zhang, Victor S Sheng, Jian Wu, Xiaoqin Fu, and Xindong Wu, "Improving label quality in crowdsourcing using noise correction," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 2015, pp. 1931–1934.

[50] Vikas C Raykar and Shipeng Yu, "Eliminating spammers and ranking annotators for crowdsourced labeling tasks," *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 491–518, 2012.

[51] Carsten Eickhoff and Arjen de Vries, "How crowdsourcable is your task," in *Proceedings of the workshop on crowdsourcing for search and data mining (CSDM) at the fourth ACM international conference on web search and data mining (WSDM)*, 2011, pp. 11–14.

[52] Julie S Downs, Mandy B Holbrook, Steve Sheng, and Lorrie Faith Cranor, "Are your participants gaming the system?: screening mechanical turk workers," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2010, pp. 2399–2402.

[53] Kuan-Ta Chen, Chen-Chi Wu, Yu-Chun Chang, and Chin-Laung Lei, "A crowdsourceable qoe evaluation framework for multimedia content," in *Proceedings of the 17th ACM international conference on Multimedia*. ACM, 2009, pp. 491–500.

[54] Michael S Bernstein, Greg Little, Robert C Miller, Björn Hartmann, Mark S Ackerman, David R Karger, David Crowell, and Katrina Panovich, "Soylent: a word processor with a crowd inside," *Communications of the ACM*, vol. 58, no. 8, pp. 85–94, 2015.

[55] John Hattie and Helen Timperley, "The power of feedback," *Review of educational research*, vol. 77, no. 1, pp. 81–112, 2007.

[56] Patrick O'Shea and Jennifer Kidd, "Crowdsourced grading: Exploring the validity and effects of student-authored and student-evaluated textbooks," in *Global Learn*. Association for the Advancement of Computing in Education (AACE), 2011, pp. 1266–1271.

[57] Ben Lorica, "Semi-automatic method for grading a million homework assignments," *Online at http://radar.oreilly.com/2013/10/semi-automatic-method-for-grading-a-million-homework-assignments.html*.

[58] Serge Dulucq and Hélène Touzet, "Analysis of tree edit distance algorithms," in *Annual Symposium on Combinatorial Pattern Matching*. Springer, 2003, pp. 83–95.

[59] Philip Bille, "A survey on tree edit distance and related problems," *Theoretical computer science*, vol. 337, no. 1-3, pp. 217–239, 2005.

[60] 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 (CIR11)*, 2011, pp. 21–26.

[61] Jeff Howe, "The rise of crowdsourcing," *Wired magazine*, vol. 14, no. 6, pp. 1–4, 2006.

[62] Luis Von Ahn, "Human computation," in *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*. IEEE Computer Society, 2008, pp. 1–2.

[63] Xuan Liu, Meiyu Lu, Beng Chin Ooi, Yanyan Shen, Sai Wu, and Meihui Zhang, "Cdas: a crowdsourcing data analytics system," *Proceedings of the VLDB Endowment*, vol. 5, no. 10, pp. 1040–1051, 2012.

[64] Nguyen Quoc Viet Hung, Nguyen Thanh Tam, Lam Ngoc Tran, and Karl Aberer, "An evaluation of aggregation techniques in crowdsourcing," in *International Conference on Web Information Systems Engineering*. Springer, 2013, pp. 1–15.

[65] Ludmila I Kuncheva, Christopher J Whitaker, Catherine A Shipp, and Robert PW Duin, "Limits on the majority vote accuracy in classifier fusion," *Pattern Analysis & Applications*, vol. 6, no. 1, pp. 22–31, 2003.

[66] Kyumin Lee, James Caverlee, and Steve Webb, "The social honeypot project: protecting online communities from spammers," in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 1139–1140.

[67] Faiza Khan Khattak and Ansaf Salleb-Aouissi, "Quality control of crowd labeling through expert evaluation," in *Proceedings of the NIPS 2nd Workshop on Computational Social Science and the Wisdom of Crowds*, 2011, vol. 2.

[68] 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)*, pp. 1–38, 1977.

[69] Padhraic Smyth, Usama M Fayyad, Michael C Burl, Pietro Perona, and Pierre Baldi, "Inferring ground truth from subjective labelling of venus images," in *Advances in neural information processing systems*, 1995, pp. 1085–1092.

[70] Rong Jin and Zoubin Ghahramani, "Learning with multiple labels," in *Advances in neural information processing systems*, 2003, pp. 921–928.

[71] 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*, vol. 11, no. Apr, pp. 1297–1322, 2010.

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

[73] Stuart Geman and Donald Geman, "Stochastic relaxation, gibbs distributions, and the bayesian restoration of images," in *Readings in Computer Vision*, pp. 564–584. Elsevier, 1987.

[74] Akash Das Sarma, Aditya Parameswaran, and Jennifer Widom, "Globally optimal crowdsourcing quality management," *arXiv preprint arXiv:1502.07710*, 2015.

[75] Fabian L Wauthier and Michael I Jordan, "Bayesian bias mitigation for crowdsourcing," in *Advances in neural information processing systems*, 2011, pp. 1800–1808.

[76] Honglei Zhuang and Joel Young, "Leveraging in-batch annotation bias for crowdsourced active learning," in *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*. ACM, 2015, pp. 243–252.

[77] John Le, Andy Edmonds, Vaughn Hester, and Lukas Biewald, "Ensuring quality in crowdsourced search relevance evaluation: The effects of training question distribution," in *SIGIR 2010 workshop on crowdsourcing for search evaluation*, 2010, vol. 2126.

[78] Kumar Abhinav, NC Shrikanth, and Anurag Dwarakanath, "Simulation of consensus based approaches to mitigate the challenges in crowdsourcing.," in *QuASoQ/WAWSE/CMCE@ APSEC*, 2015, pp. 49–51.

[79] Jeroen BP Vuurens, Carsten Eickhoff, and Arjen P de Vries, "Managing the quality of large-scale crowdsourcing.," in *TREC*. Citeseer, 2011.

[80] Bob Carpenter, "Multilevel bayesian models of categorical data annotation," *Unpublished manuscript*, vol. 17, no. 122, pp. 45–50, 2008.

[81] Jing Wang, Panagiotis G Ipeirotis, and Foster Provost, "Managing crowdsourcing workers," in *The 2011 winter conference on business intelligence*. Citeseer, 2011, pp. 10–12.

[82] Aniket Kittur, Ed H Chi, and Bongwon Suh, "Crowdsourcing user studies with mechanical turk," in *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, 2008, pp. 453–456.

[83] Zoubin Ghahramani and Thomas L Griffiths, "Infinite latent feature models and the indian buffet process," in *Advances in neural information processing systems*, 2006, pp. 475–482.

[84] Haoqi Zhang, Eric Horvitz, Rob C Miller, and David C Parkes, "Crowdsourcing general computation," 2011.

[85] Michael Isard, Mihai Budiu, Yuan Yu, Andrew Birrell, and Dennis Fetterly, "Dryad: distributed data-parallel programs from sequential building blocks," in *ACM SIGOPS operating systems review*. ACM, 2007, vol. 41, pp. 59–72.

[86] Christopher Olston, Benjamin Reed, Utkarsh Srivastava, Ravi Kumar, and Andrew Tomkins, "Pig latin: a not-so-foreign language for data processing," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. ACM, 2008, pp. 1099–1110.

[87] Jeffrey Dean and Sanjay Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[88] Eric J Horvitz, "Problem-solving design: Reasoning about computational value, tradeoffs, and resources," in *Proceedings of the Second Annual NASA Research Forum*, 1987, pp. 26–43.

[89] Douglas R Smith, "Top-down synthesis of divide-and-conquer algorithms," in *Readings in artificial intelligence and software engineering*, pp. 35–61. Elsevier, 1986.

[90] Anbang Xu, Shih-Wen Huang, and Brian Bailey, "Voyant: generating structured feedback on visual designs using a crowd of non-experts," in *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*. ACM, 2014, pp. 1433–1444.

[91] Kurt Luther, Jari-Lee Tolentino, Wei Wu, Amy Pavel, Brian P Bailey, Maneesh Agrawala, Björn Hartmann, and Steven P Dow, "Structuring, aggregating, and evaluating crowdsourced design critique," in *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*. ACM, 2015, pp. 473–485.

[92] Michael D Greenberg, Matthew W Easterday, and Elizabeth M Gerber, "Critiki: A scaffolded approach to gathering design feedback from paid crowdworkers," in *Proceedings of the 2015 ACM SIGCHI Conference on Creativity and Cognition*. ACM, 2015, pp. 235–244.

[93] Hanover Research, "Effective grading practices in the middle school and high school environments," *Online at http://docplayer.net/5893105-Effective-grading-practices-in-the-middle-school-and-high-school-environments.html*.

[94] James S Terwilliger, "4. some thoughts on grading systems and grading practices," 1993.

[95] James D Allen, "Grades as valid measures of academic achievement of classroom learning," *The Clearing House: A Journal of Educational Strategies, Issues and Ideas*, vol. 78, no. 5, pp. 218–223, 2005.

[96] Arlene A Russell, "Calibrated peer review-a writing and critical-thinking instructional tool," *Teaching Tips: Innovations in Undergraduate Science Instruction*, vol. 54, 2004.

[97] Philip M Sadler and Eddie Good, "The impact of self-and peer-grading on student learning," *Educational assessment*, vol. 11, no. 1, pp. 1–31, 2006.

[98] Fiona Godlee and Tom Jefferson, *Peer review in health sciences*, BMJ Books London, 2003.

[99] Philip M Sadler and Eddie Good, "The impact of self-and peer-grading on student learning," *Educational assessment*, vol. 11, no. 1, pp. 1–31, 2006.

[100] Andrew Gelman and Jennifer Hill, *Data analysis using regression and multilevel/hierarchical models*, Cambridge university press, 2006.

[101] Nihar B Shah, Joseph K Bradley, Abhay Parekh, Martin Wainwright, and Kannan Ramchandran, "A case for ordinal peer-evaluation in moocs," in *NIPS Workshop on Data Driven Education*, 2013.

[102] Karthik Raman and Thorsten Joachims, "Methods for ordinal peer grading," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 1037–1046.

[103] Richard Smith, "Peer review: a flawed process at the heart of science and journals," *Journal of the royal society of medicine*, vol. 99, no. 4, pp. 178–182, 2006.

[104] Xi Chen, Paul N Bennett, Kevyn Collins-Thompson, and Eric Horvitz, "Pairwise ranking aggregation in a crowdsourced setting," in *Proceedings of the sixth ACM international conference on Web search and data mining*. ACM, 2013, pp. 193–202.

[105] Luis Von Ahn and Laura Dabbish, "Labeling images with a computer game," in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 2004, pp. 319–326.

[106] Barzan Mozafari, Purnamrita Sarkar, Michael J Franklin, Michael I Jordan, and Samuel Madden, "Active learning for crowd-sourced databases," *arXiv preprint arXiv:1209.3686*, 2012.

[107] Ido Dagan, Oren Glickman, and Bernardo Magnini, "The pascal recognising textual entailment challenge," in *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, pp. 177–190. Springer, 2006.

[108] Sameer S Pradhan, Edward Loper, Dmitriy Dligach, and Martha Palmer, "Semeval-2007 task 17: English lexical sample, srl and all words," in *Proceedings of the 4th International Workshop on Semantic Evaluations*. Association for Computational Linguistics, 2007, pp. 87–92.

[109] Edith Law and Luis Von Ahn, "Input-agreement: a new mechanism for collecting data using human computation games," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2009, pp. 1197–1206.

[110] Christian Körner and Markus Strohmaier, "A call for social tagging datasets," *ACM SIGWEB Newsletter*, , no. Winter, pp. 2, 2010.

[111] Panos Ipeirotis, "Mechanical turk: Now with 40.92% spam," *Behind Enemy Lines blog*, 2010.

[112] Chris Buckley, Matthew Lease, Mark D Smucker, Hyun Joon Jung, Catherine Grady, Chris Buckley, Matthew Lease, Mark D Smucker, Catherine Grady, Matthew Lease, et al., "Overview of the trec 2010 relevance feedback track (notebook)," in *The nineteenth text retrieval conference (TREC) notebook*, 2010.

[113] Hyun Joon Jung and Matthew Lease, "Improving quality of crowdsourced labels via probabilistic matrix factorization," in *Proceedings of the 4th Human Computation Workshop (HCOMP) at AAAI*, 2012, pp. 101–106.

[114] Upwork, "Upwork, the worlds largest online workplace," *Online at https://www.upwork.com*, 2015.

[115] Christopher J Pannucci and Edwin G Wilkins, "Identifying and avoiding bias in research," *Plastic and reconstructive surgery*, vol. 126, no. 2, pp. 619, 2010.

[116] Stan Z Li and Anil K Jain, *Encyclopedia of Biometrics: I-Z.*, vol. 1, Springer Science & Business Media, 2009.

[117] Tianfeng Chai and Roland R Draxler, "Root mean square error (rmse) or mean absolute error (mae)?–arguments against avoiding rmse in the literature," *Geoscientific model development*, vol. 7, no. 3, pp. 1247–1250, 2014.

[118] Luis Von Ahn, Benjamin Maurer, Colin McMillen, David Abraham, and Manuel Blum, "recaptcha: Human-based character recognition via web security measures," *Science*, vol. 321, no. 5895, pp. 1465–1468, 2008.

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

[120] Ofer Dekel and Ohad Shamir, "Vox populi: Collecting high-quality labels from a crowd," 2009.

[121] Gianluca Demartini, Djellel Eddine Difallah, and Philippe Cudré-Mauroux, "Zencrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking," in *Proceedings of the 21st international conference on World Wide Web*. ACM, 2012, pp. 469–478.

[122] Xin Luna Dong, Laure Berti-Equille, and Divesh Srivastava, "Integrating conflicting data: the role of source dependence," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 550–561, 2009.

[123] Seyda Ertekin, Haym Hirsh, and Cynthia Rudin, "Learning to predict the wisdom of crowds," *arXiv preprint arXiv:1204.3611*, 2012.

[124] Alban Galland, Serge Abiteboul, Amélie Marian, and Pierre Senellart, "Corroborating information from disagreeing views," in *Proceedings of the third ACM international conference on Web search and data mining*. ACM, 2010, pp. 131–140.

[125] Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin, *Bayesian data analysis*, CRC press, 2013.

[126] Uri Gneezy and Aldo Rustichini, "Pay enough or don't pay at all," *The Quarterly Journal of Economics*, vol. 115, no. 3, pp. 791–810, 2000.

[127] Jeff Pasternack and Dan Roth, "Knowing what to believe (when you already know something)," in *Proceedings of the 23rd International Conference on Computational Linguistics*. Association for Computational Linguistics, 2010, pp. 877–885.

[128] Chuishi Meng, Wenjun Jiang, Yaliang Li, Jing Gao, Lu Su, Hu Ding, and Yun Cheng, "Truth discovery on crowd sensing of correlated entities," in *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2015, pp. 169–182.

[129] Rebecca J Passonneau and Bob Carpenter, "The benefits of a model of annotation," *Transactions of the Association for Computational Linguistics*, vol. 2, pp. 311–326, 2014.

[130] Gabriella Kazai, Jaap Kamps, and Natasa Milic-Frayling, "Worker types and personality traits in crowdsourcing relevance labels," in *Proceedings of the 20th ACM international conference on Information and knowledge management*. ACM, 2011, pp. 1941–1944.

[131] Denny 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*, 2012, pp. 2195–2203.

[132] Qi Li, Yaliang Li, Jing Gao, Bo Zhao, Wei Fan, and Jiawei Han, "Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation," in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. ACM, 2014, pp. 1187–1198.

[133] Jing Zhang, Victor S Sheng, Qianmu Li, Jian Wu, and Xindong Wu, "Consensus algorithms for biased labeling in crowdsourcing," *Information Sciences*, vol. 382, pp. 254–273, 2017.

[134] Panagiotis G Ipeirotis and Evgeniy Gabrilovich, "Quizz: targeted crowdsourcing with a billion (potential) users," in *Proceedings of the 23rd international conference on World wide web*. ACM, 2014, pp. 143–154.

[135] Dimitri P Bertsekas, *Nonlinear programming*, Athena scientific Belmont, 1999.

[136] Omar Alonso, Daniel E Rose, and Benjamin Stewart, "Crowdsourcing for relevance evaluation," in *ACM SigIR Forum*. ACM, 2008, vol. 42, pp. 9–15.

[137] Jeroen BP Vuurens and Arjen P de Vries, "Obtaining high-quality relevance judgments using crowdsourcing," *IEEE Internet Computing*, vol. 16, no. 5, pp. 20–27, 2012.

[138] Hiroshi Kajino, Yuta Tsuboi, and Hisashi Kashima, "Clustering crowds," in *AAAI*, 2013.

[139] Thomas W Malone and Kevin Crowston, "The interdisciplinary study of coordination," *ACM Computing Surveys (CSUR)*, vol. 26, no. 1, pp. 87–119, 1994.

[140] Anand Kulkarni, Matthew Can, and Björn Hartmann, "Collaboratively crowdsourcing workflows with turkomatic," in *Proceedings of the acm 2012 conference on computer supported cooperative work*. ACM, 2012, pp. 1003–1012.

[141] Nigel P Melville, "Crowd-sourced peer feedback (cpf) for learning community engagement: Results and reflections from a pilot study," in *System Sciences (HICSS), 2014 47th Hawaii International Conference on*. IEEE, 2014, pp. 32–41.

[142] Shayan Doroudi, Ece Kamar, Emma Brunskill, and Eric Horvitz, "Toward a learning science for complex crowdsourcing tasks," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 2016, pp. 2623–2634.

[143] David Boud, *Enhancing learning through self-assessment*, Routledge, 2013.

[144] Maddalena Taras, "Using assessment for learning and learning from assessment," *Assessment & Evaluation in Higher Education*, vol. 27, no. 6, pp. 501–510, 2002.

[145] Avrim Blum and Tom Mitchell, "Combining labeled and unlabeled data with co-training," in *Proceedings of the eleventh annual conference on Computational learning theory*. ACM, 1998, pp. 92–100.

[146] Kamal Nigam and Rayid Ghani, "Analyzing the effectiveness and applicability of co-training," in *Proceedings of the ninth international conference on Information and knowledge management*. ACM, 2000, pp. 86–93.

[147] Steffen Bickel and Tobias Scheffer, "Multi-view clustering," in *ICDM*, 2004, vol. 4, pp. 19–26.

[148] Justin Cheng, Jaime Teevan, Shamsi T Iqbal, and Michael S Bernstein, "Break it down: A comparison of macro-and microtasks," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 2015, pp. 4061–4064.

[149] Haoqi Zhang, Edith Law, Rob Miller, Krzysztof Gajos, David Parkes, and Eric Horvitz, "Human computation tasks with global constraints," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2012, pp. 217–226.

[150] Walter Lasecki, Christopher Miller, Adam Sadilek, Andrew Abumoussa, Donato Borrello, Raja Kushalnagar, and Jeffrey Bigham, "Real-time captioning by groups of non-experts," in *Proceedings of the 25th annual ACM symposium on User interface software and technology*. ACM, 2012, pp. 23–34.

[151] Greg Little, Lydia B Chilton, Max Goldman, and Robert C Miller, "Turkit: human computation algorithms on mechanical turk," in *Proceedings of the 23nd annual ACM symposium on User interface software and technology*. ACM, 2010, pp. 57–66.

[152] Salman Ahmad, Alexis Battle, Zahan Malkani, and Sepander Kamvar, "The jabberwocky programming environment for structured social computing," in *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, 2011, pp. 53–64.

[153] David Oleson, Alexander Sorokin, Greg P Laughlin, Vaughn Hester, John Le, and Lukas Biewald, "Programmatic gold: Targeted and scalable quality assurance in crowdsourcing," *Human computation*, vol. 11, no. 11, 2011.

[154] Wesley Willett, Jeffrey Heer, and Maneesh Agrawala, "Strategies for crowdsourcing social data analysis," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2012, pp. 227–236.

[155] Mira Dontcheva, Robert R Morris, Joel R Brandt, and Elizabeth M Gerber, "Combining crowdsourcing and learning to improve engagement and performance," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2014, pp. 3379–3388.

[156] Adish Singla, Ilija Bogunovic, Gábor Bartók, Amin Karbasi, and Andreas Krause, "Near-optimally teaching the crowd to classify.," in *ICML*, 2014, pp. 154–162.

[157] Tanushree Mitra, Clayton J Hutto, and Eric Gilbert, "Comparing person-and process-centric strategies for obtaining quality data on amazon mechanical turk," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 2015, pp. 1345–1354.

[158] Ron JCM Salden, Kenneth R Koedinger, Alexander Renkl, Vincent Aleven, and Bruce M McLaren, "Accounting for beneficial effects of worked examples in tutored problem solving," *Educational Psychology Review*, vol. 22, no. 4, pp. 379–392, 2010.

[159] John Sweller and Graham A Cooper, "The use of worked examples as a substitute for problem solving in learning algebra," *Cognition and instruction*, vol. 2, no. 1, pp. 59–89, 1985.

[160] Fleurie Nievelstein, Tamara Van Gog, Gijs Van Dijck, and Henny PA Boshuizen, "The worked example and expertise reversal effect in less structured tasks: Learning to reason about legal cases," *Contemporary Educational Psychology*, vol. 38, no. 2, pp. 118–125, 2013.

[161] Chinmay Kulkarni, Koh Pang Wei, Huy Le, Daniel Chia, Kathryn Papadopoulos, Justin Cheng, Daphne Koller, and Scott R Klemmer, "Peer and self assessment in massive online classes," in *Design thinking research*, pp. 131–168. Springer, 2015.

[162] Hyun Joon Jung and Matthew Lease, "A discriminative approach to predicting assessor accuracy," in *European Conference on Information Retrieval*. Springer, 2015, pp. 159–171.

[163] Pinar Donmez, Jaime Carbonell, and Jeff Schneider, "A probabilistic framework to learn from multiple annotators with time-varying accuracy," in *Proceedings of the 2010 SIAM International Conference on Data Mining*. SIAM, 2010, pp. 826–837.

[164] 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*. ACM, 2010, pp. 557–566.

[165] Alexander Sorokin and David Forsyth, "Utility data annotation with amazon mechanical turk," in *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on*. IEEE, 2008, pp. 1–8.

[166] Omar Alonso and Stefano Mizzaro, "Can we get rid of trec assessors? using mechanical turk for relevance assessment," in *Proceedings of the SIGIR 2009 Workshop on the Future of IR Evaluation*, 2009, vol. 15, p. 16.

[167] Vamshi Ambati, Stephan Vogel, and Jaime G Carbonell, "Active learning and crowd-sourcing for machine translation," 2010.

[168] Michael Prince, "Does active learning work? a review of the research," *Journal of engineering education*, vol. 93, no. 3, pp. 223–231, 2004.

[169] Liyue Zhao, Gita Sukthankar, and Rahul Sukthankar, "Incremental relabeling for active learning with noisy crowdsourced annotations," in *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on*. IEEE, 2011, pp. 728–733.

[170] Joana Costa, Catarina Silva, Mário Antunes, and Bernardete Ribeiro, "On using crowdsourcing and active learning to improve classification performance," in *Intelligent Systems Design and Applications (ISDA), 2011 11th International Conference on*. IEEE, 2011, pp. 469–474.

[171] Meng Fang, Jie Yin, and Dacheng Tao, "Active learning for crowdsourcing using knowledge transfer," in *AAAI*, 2014, pp. 1809–1815.

[172] Jing Zhang, Xindong Wu, and Victor S Shengs, "Active learning with imbalanced multiple noisy labeling," *IEEE transactions on cybernetics*, vol. 45, no. 5, pp. 1095–1107, 2015.

[173] P Ipeirotis, "Worker evaluation in crowdsourcing: Gold data or multiple workers," 2010.

[174] Guo Xintong, Wang Hongzhi, Yangqiu Song, and Gao Hong, "Brief survey of crowdsourcing for data mining," *Expert Systems with Applications*, vol. 41, no. 17, pp. 7987–7994, 2014.

[175] Michael S Bernstein, Jaime Teevan, Susan Dumais, Daniel Liebling, and Eric Horvitz, "Direct answers for search queries in the long tail," in *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, 2012, pp. 237–246.

[176] Tobias Hoßfeld, Michael Seufert, Matthias Hirth, Thomas Zinner, Phuoc Tran-Gia, and Raimund Schatz, "Quantification of youtube qoe via crowdsourcing," in *Multimedia (ISM), 2011 IEEE International Symposium on*. IEEE, 2011, pp. 494–499.

[177] Jean-Francois Paiement, James G Shanahan, and Remi Zajac, "Crowdsourcing local search relevance," *Proceedings of the CrowdConf 2010*, 2010.

[178] Zhiquan Liu, Luo Luo, and Wu-Jun Li, "Robust crowdsourced learning," in *Big Data, 2013 IEEE International Conference on*. IEEE, 2013, pp. 338–343.

[179] Hyun Joon Jung, Yubin Park, and Matthew Lease, "Predicting next label quality: A time-series model of crowdwork," in *Second AAAI Conference on Human Computation and Crowdsourcing*, 2014.

[180] Matthias Hirth, Tobias Hoßfeld, and Phuoc Tran-Gia, "Cheat-detection mechanisms for crowdsourcing," *University of Würzburg, Tech. Rep*, vol. 4, 2010.

[181] Eric Huang, Haoqi Zhang, David C Parkes, Krzysztof Z Gajos, and Yiling Chen, "Toward automatic task design: a progress report," in *Proceedings of the ACM SIGKDD workshop on human computation*. ACM, 2010, pp. 77–85.

CURRICULUM VITAE

NAME: Lingyu Lyu

ADDRESS: Computer Engineering & Computer Science Department
Speed School of Engineering
University of Louisville
Louisville, KY 40292

EDUCATION:

Ph.D., Computer Science & Engineering
August 2018
University of Louisville, *Louisville, Kentucky*

Graduate Certificate in Data Science
December 2016
University of Louisville, *Louisville, Kentucky*

M.S., Computer Science & Engineering
August 2014
University of Louisville, *Louisville, Kentucky*

B.Tech., Computer Science
June 2011
Soochow University, China

PUBLICATIONS:

1. **Lyu, L.**, Kantardzic, M., and Sethi, T.S. Sloppiness mitigation in crowdsourcing: detecting and correcting bias for crowd scoring tasks. Data Science and Analytics. Springer, 2018.

2. Sethi, T.S., Kantardzic, M., **Lyu, L.**, and Chen J. A dynamic-adversarial mining approach to

the security of machine learning. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. Wiley, 2018.

3. **Lyu., L.**, Kantardzic, M., and Hu, H. Work like an expert: an integrated multiple views approach for crowdsourcing complex tasks. Manuscript submitted for publication, 2018.

4. **Lyu, L.**, and Kantardzic, M. Evaluating complex tasks through crowdsourcing: multiple views approach. ICCSE, 2016.

5. **Lyu, L.**, Kantardzic, M., and Arabmakki, E. Solar irradiance forecasting by using wavelet based denoising. Proceeding in IEEE Symposium Series on Computational Intelligence (SSCI). IEEE, 2014.

PROFESSIONAL EXPERIENCE

- Mobile App Developer Coop, Kindred Healthcare, Louisville, KY, USA

  Developed cross-platform mobile application for Kindred Healthcare. Addressed platform specific problems such as keyboard, screen size and alignment of the interface, and utilized RESTful web service to retrieve data from back-end.

  August 2016 - January 2017

- Software Engineer Intern, The Rawlings Group, LaGrange, KY, USA

  Developed a Windows service which automatically retrieves and executes SQL script on servers for QAs. Worked on developing multiple web applications and user interfaces, as well as the SSIS projects.

  May 2015 - August 2015

- Ph.D. Research Experience

  Research in truth discovery for Crowdsourcing systems, time series data predicting, streaming data mining, Bayesian modeling, and distributed data mining using Hadoop, Spark, and MapReduce.

  August 2012 - August 2018

- Graduate Teaching Assistant, University of Louisville, Louisville, Kentucky, USA

  Graduate teaching assistant for Data Mining course in Fall 2017, Simulation of Discrete Systems course in Summer 2016/2017, Introduction to Cryptography course in Spring 2017, and

Object Oriented Programming with Java in Summer 2016. Responsible for course delivery, grading, projects creation, and problem solving for students.

August 2014 - August 2018

HONORS AND AWARDS:

1. CSE Doctoral Award, Departmental Award, 2018
2. Doctoral Dissertation Completion Award, University of Louisville, 2017
3. Graduate Research Assistant Scholarship, University of Louisville, 2016-2017
4. University of Louisville Tuition Award, 2012-2016
5. Academic Scholarship, Beihang University, China, 2011-2012